uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Hoi Ting Poon

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.A.Sc. (Electrical Engineering)

GRADE / DEGREE

Department of Electrical Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Towards a Practical and Secure Biometric Authentication System

TITRE DE LA THÈSE / TITLE OF THESIS

Prof. A. Miri

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Prof. J. Zhao

Prof. I. Marsland

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

Towards a Practical and Secure Biometric Authentication System

A Thesis Submitted to
The Faculty of Graduate and Postdoctoral Studies
in Partial Fulfillment of The Requirements
for The Degree of
Master of Applied Science
in
Electrical and Computer Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering
School of Information Technology and Engineering
University of Ottawa

Poon, Hoi Ting
January 2008

Your file Votre référence
ISBN: 978-0-494-50917-3
Our file Notre référence
ISBN: 978-0-494-50917-3

Abstract

The idea of merging biometrics technology with cryptography brought interesting possibilities in enhancing the security and privacy of biometrics systems. Conventional systems generally require large databases, which represent a security risk and raise privacy concerns. Biometric encryption is a method devised to hide biometric features along with a cryptographic key, which could remove the need for such biometric databases. The Fuzzy Vault scheme is one of the promising candidates for biometric encryption. This thesis analyzes the scheme and attempts to provide solutions to problems impeding the system from being used in practise. We looked at the current techniques in dealing with the noise inherent in biometric templates, in particular fingerprints, and provides insight on their implication in information and biometric security. We also present two practical decoders based on the Berlekamp-Massey algorithm and the Euclidean algorithm, and provide the first implementation of a Reed-Solomon decoder for the Fuzzy Vault scheme. Our implementation results indicate that the traditional Berlekamp-Massey algorithm may not be as suitable and efficient as Gao's Reed-Solomon decoder. In our analysis, some potential vulnerabilities were also identified. In particular, the collusion attack was found to be able to seriously reduce the security of the scheme in practise and is applicable to all existing implementations. Some possible defenses against the attack such as a one-way transform of the locking set and deterministic chaff points generation were proposed.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Some would say we are inching towards a surveillance society. Since the events of September 11^{th} 2001, security has become a top priority for many countries. Governments around the world embrace biometrics as a mean to control access in the hope of averting similar tragedies. In airports and border crossings, biometric identification cards are now commonly used. Several countries have unveiled plans to include biometrics in passports, national ID cards and even driver licenses. In the US, the FBI's Integrated Automated Fingerprint Identification System maintains the world's largest fingerprint database, with over 47 million subjects. Even in our every day life, biometrics seem to have taken hold. You can pay for groceries, gas and even borrow books with the touch of a finger. Those in favor of biometrics say that the technology enhances security and enables businesses to provide more efficient services. Others worry that personal information may be misused and the technology would lead us towards an Orwellian state. Is it possible for biometrics to be used such that the privacy of the users remains protected?

Biometric systems generally require the storage of biometric information in large databases. These centralized servers are often a cause for privacy concerns. A breach in the system would be catastrophic, as a person's biometrics cannot be replaced. In an attempt to enhance privacy in biometric systems, researchers recently began to investigate the potential benefit of merging biometric technology with cryptography. These techniques are often called biometric encryption. The goal is to allow for a secret to be encrypted using a set of biometric features, such that neither the features nor the secret need to be stored in easily accessible form. Figure 1.1 shows how biometric

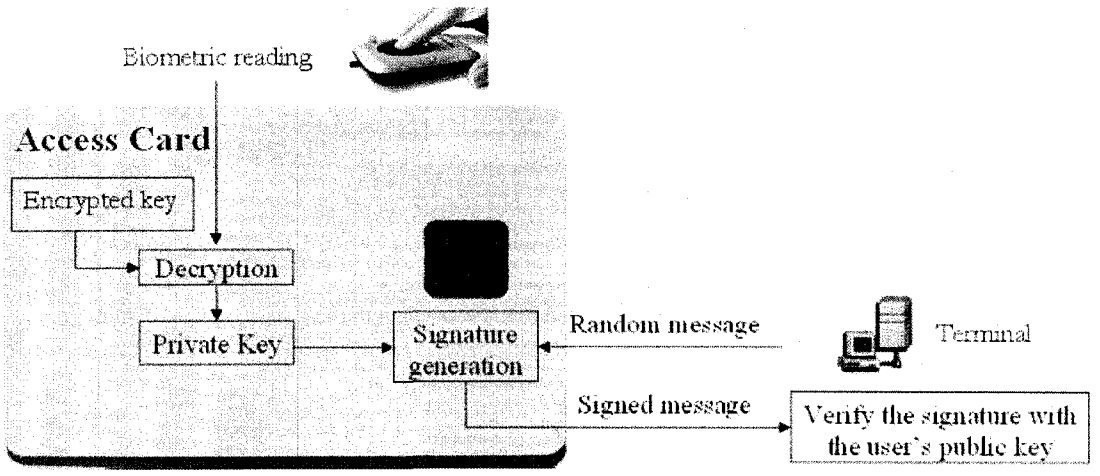

Figure 1.1: The use of biometric encryption in an authentication system with smart cards[18]

encryption can be used for authentication. The secret and the biometric are both in encrypted form. Hence, they no longer need to be stored in biometric databases. Instead, smart cards can be used and a standard public key based signature scheme serve to authenticate users [18].

In cryptography, messages are hidden and reveal by means of a cryptographic key. Unless the exact key is used during decryption, the message cannot be revealed. However, biometric impressions are always different due to elasticity, aging and various variability in the capturing process. One of the main challenges is then to devise a cryptographic scheme which would allow for such variability in the key.

In 1999, Juel and Wattenberg [9] introduced a cryptographic primitive called Fuzzy Commitment scheme to address the problem of secure biometric storage. The conceptually simple scheme makes use of error correction codes to allow for error tolerance in the key. In another word, the scheme allows a user to encrypt a message using a key K and decrypt the message with a key K' within a certain distance from K. This error tolerance capability allows for many interesting applications such as privacy-protected matching and the use of biometric keys in cryptosystems[8]. However, the scheme has an important shortcoming. Geometric distortions such as rotations and translations are common in biometric readings. Hence, the key K' will likely consist of some permutation of the elements in K. The Fuzzy Commitment scheme does not tolerate such

re-ordering of the symbols in the key [8].

To address this problem, Juel and Sudan [8] proposed the Fuzzy Vault scheme. By characterising the key as a set of symbols instead of a sequence and combining with the proper error correction code, the scheme achieves the property of order-invariance. Recognizing its potential as a replacement for key release based biometrics systems, several researchers [18, 19] and [23] have already provided implementations of the Fuzzy Vault scheme, in particular for fingerprints. However, several problems must be solved before the scheme can be used in practise. The two main difficulties pertaining to Fuzzy Vault for fingerprints are the alignment problem and the decoding problem. The alignment problem refers to the difficulty in aligning the live template with the enrolled template due to geometric distortion. Most of the current implementations consist of attempts to solve this problem through techniques such as quantization, lattices and rotation-translation invariant representations. The decoding problem refers to the task of recovering the message given the key and the ciphertext. In the original paper by Sudan [8], the decoding problem was briefly stated as possible without demonstration through the use of Berlekamp-Massey algorithm. Some researchers have since doubted the viability of a Reed-Solomon decoder for Fuzzy Vault and used an alternative decoding method [10, 19, 21]. To the best of our knowledge, no Reed-Solomon decoder has yet to be implemented for Fuzzy Vault. Despite numerous implementations of the Fuzzy Vault scheme for biometrics, the security analysis is usually brief and based on the assumption that the most efficient method to discover the message is a brute-force attack.

In this thesis, we present two decoding algorithms for Fuzzy Vault based on the original interpretations of Reed-Solomon codes and describe a potentially serious vulnerability in the scheme applicable to most existing implementations. The discussion begins by describing the Fuzzy Vault algorithms and the information theoretic security. The typical key release based biometric system and the advantages of a key generation system are then presented in chapter 3, along with the potential use of Fuzzy Vault as a key generation system explained through fingerprints. In chapter 4, we discuss the CRC decoding algorithms used in most of the current implementations and the need for a practical decoder. To provide the background needed for the proposed decoding algorithms, the chapter continues with a discussion on Reed-Solomon codes and its role in Fuzzy Vault. The proposed Reed-Solomon decoding algorithms based on the

Berlekamp-Massey algorithm [15] and Gao's algorithm [5] are then presented. In chapter 5, some potential vulnerabilities, in particular the collusion attack, are discussed and some modifications are suggested. Finally, we conclude with some suggestions on topics for future research.

Chapter 2

Related Work

A fuzzy commitment scheme, introduced by Juel and Wattenberg, is a cryptosystem in which errors in the encryption key can be tolerated through the use of error correction codes. Its construction is conceptually simple and enjoys provable security [9]. A secret codeword, c, is encrypted using the key, x, as $v = c - x$. The decrypted codeword using the key, x', is $c' = f(v' + x')$, where $f(b)$ maps b to the nearest codeword. If the distance between x and x' is small, the decryption will be successful, $c' = c$. Although this scheme tolerates some errors in the information symbols in the codeword, it does not tolerate substantial reordering of these symbols. Since biometric information is often affected by translation, rotation and distortion errors, x' will likely consists of a permutation of the symbols in x. If a significant amount of symbols in x do not line up with the corresponding symbols in x', the error correction will surely fail. Hence, the property of order-invariance is crucial to the functioning of the scheme for biometrics [9]. To address the need for order-invariance, Juel and Sudan proposed the Fuzzy Vault scheme.

In this chapter, we present the Fuzzy Vault algorithms and relevant works on implementations using fingerprints. A brief discussion on the security of the scheme is also presented.

2.1 The Fuzzy Vault scheme

Traditionally, cryptographic keys have always been exact. Unless the user has the key to decrypt the message, he will not learn anything about the message. Thus, errors cannot

be tolerated in the keys. However, there are situations where error tolerance would be beneficial. For example, Alice wishes to meet someone who has similar interests. If Bob's interests overlaps significantly with Alice's, she would surely like to meet him despite not having exactly the same set of interests. In [9], a class of attributes-based cryptosystems was introduced, which allows for errors in the keys. In these systems, we view the key as a set of attributes (e.g. Name, age, employee number). In the previous example, we may have Alice's contact information, encrypted with her set of interests. Hence, unless Bob shares many similar interests, he would not be able to contact Alice [8]. Attributes-based cryptography can also be used to incorporate biometrics, where each reading contains a certain amount of errors.

2.1.1 The algorithm

The Fuzzy Vault scheme contains two main algorithms: *Lock*, *Unlock*.

Setup

For the functionality of the Fuzzy Vault system, one must first decide on a field $\mathbb{F}$ of order q, where q depends on the required security.

Lock

The *Lock* algorithm [8] is analogous to the encryption algorithm. Given a locking set (e.g. biometrics key) $A = \{a_i\}_{i=1}^{t}$, where $a_i \in \mathbb{F}$, and a message, $M = \{m_i\}_{i=1}^{k}$, where $m_i \in \mathbb{F}$, to be locked, generates a set $V_A = \{(x_i, y_i)\}_{i=1}^{r}$, where $x_i, y_i \in \mathbb{F}$:

- Create two empty sets, V_A and X
- Set $Y(x) = m_1 + m_2 x + m_3 x^2 + \ldots m_k x^{k-1}$
- For $i = 1$ to t
  - $X = X \cup a_i$
  - $V_A = V_A \cup (a_i, Y(a_i))$
- For $i = t + 1$ to r
  - $x_i = U(F - X)$, where U randomly chooses an element of the field $\mathbb{F}$ which is not part of X

- $X = X \cup x_i$
- $y_i = U(F - \{Y(x_i)\})$, where U randomly chooses an element of the field $\mathbb{F}$ which is not equal to $Y(x_i)$
- $V_A = V_A \cup (x_i, y_i)$

The resulting set V_A is called the Fuzzy Vault, locked under the set A.

As illustrated, the message is mapped to the coefficients of a polynomial, $Y(x)$, of degree $k-1$. The image of the locking set, $Y(A)$, is computed, producing a set of points $(a_i, Y(a_i))$ which lie on the polynomial $Y(x)$. To secure the message, we then add $r-t$ chaff points which are not part of the locking set and do not correspond to pairs that can be generated by $Y(x)$. Hence, any chaff point would not be a valid point on $Y(x)$ and cannot be confused with a point in the locking set. Note that q represents the size of the alphabet, r represents the size of the vault, t is the size of locking set and k is the minimum set overlap to achieve a match, where $q \geq r \geq t \geq k$. Since any sets of $k+1$ points among the t points on $Y(x)$ will allow reconstruction of the polynomial, the order of the points does not matter. The vault size is maximal when it is equal to the size of the key space. In the description of the Fuzzy Vault scheme, the size of the key space is equal to the field size. However, in practise, the key space may be less than the field size. Without loss of generalization, the key space will be assumed to be equal to the field size for the remainder of the thesis unless otherwise stated.

Unlock

The *Unlock* algorithm [8] is analogous to the decryption algorithm. Given a unlocking set $B = \{b_i\}_{i=1}^{t'}$, where $b_i \in \mathbb{F}$, and a vault, $V_A = \{(x_i, y_i)\}_{i=1}^{r}$, reproduce the message $M = \{m_i\}_{i=1}^{k}$ if $A \cap B \geq k$:

- Generate an empty set Q
- For $i = 1$ to t'
  - If $x_j = b_i$, $Q = Q \cup (x_j, y_i)$
- $M = Decode\,(Q)$

The unlocking set B will contain some correct points and some chaff points. The Unlock algorithm identifies and retains the points in the vault where the x_i values match the unlocking set elements. If the sets A and B overlap significantly, the set Q will contain mostly the correct points that lie on the polynomial $Y(x)$. *Decode*(Q) is a decoding operation that reconstructs the polynomial, $Y(x)$, based on the t' or less overlapping points. More precisely, to reconstruct a polynomial of degree $k-1$, we need to have at least k points that lie on the polynomial. Hence, to reconstruct $Y(x)$, $A \cap B \geq k$. It was suggested that *Decode*(Q) be implemented as a Reed Solomon decoder since the set Q can be considered as a generalized Reed Solomon codeword [8][14]. However, in the context of the Fuzzy Vault scheme, a RS code is not characterized in its popular form, using a generator polynomial, and the details of its construction were not demonstrated in the paper. When using the classic Peterson-Berlekamp-Massey algorithm, the condition for successful decoding the message becomes $A \cap B \geq (k+t')/2$ [8].

A simple example of the algorithm is as follows: Let the message to be encrypted be $M = (1, 3, 2)$ and Alice's locking set be $A = (2, 5, 3, 1)$. Then, $Y(x) = 1 + 3x + 2x^2$. We first compute $Y(A)$, producing the set of points $(2, 15)$, $(5, 66)$, $(3, 28)$, $(1, 6)$. Two chaff points $(4, 5)$, $(6, 50)$ are then added to construct the vault, $R = \{(1, 6)$, $(2, 15)$, $(3, 28)$, $(4, 5)$, $(5, 66)$, $(6, 50)\}$, where $Y(4) \neq 5$ and $Y(6) \neq 50$. If Bob provides the unlocking set $B = (4, 2, 3, 5)$ to unlock the vault, the algorithm would attempt to reconstruct $Y(x)$, given $Q = \{(2, 15)$, $(3, 28)$, $(4, 5)$, $(5, 66)\}$, and succeed because A and B overlaps on more than $k = 3$ points. However, if Bob provides an unlocking set B = (2,4,3,6,7), he would fail to unlock the message since the set overlaps on only 2 points [19].

Previous work on Fuzzy Vault for fingerprints

Since the publication of Juel and Sudan's work, several researchers have provided implementations of fuzzy vaults for biometrics. In particular, Clancy proposed a smart card based fingerprint authentication scheme using Fuzzy Vault [18]. In this system, a user attempts to authenticate himself to a server using his fingerprint and his smart card, which contains his encrypted private key. The server issues a random message to the user to sign. If the fingerprint matches the one used to encrypt the private key, the smart card will be able to correctly retrieve the key and sign the random message.

Using the user's public key, the server can then verify the signature and authenticate the user. In his construction of Fuzzy Vault, Clancy used the minutiae coordinates as the attributes in the biometric key. A database of pre-aligned fingerprints was used for testing. Chaff points were generated with the assumption that noise is uniformly Gaussian and a decoding scheme was not implemented.

Building upon Clancy's work, Uludag proposed a few changes to the Fuzzy Vault scheme for fingerprints [19]. Instead of storing the minutiae coordinates directly, they are first quantized according to a rectangular lattice. This accounts for the noise tolerance. For the decoding scheme, Uludag has stated that a RS decoder may not be realizable. The combination of a CRC and Lagrange interpolation was proposed as an alternative method for identifying the correct secret during decoding. Experimental results were obtained using a small database of 100 users and pre-aligned fingerprints.

Since fingerprint images are susceptible to rotation and translation errors, two impressions from the same finger are rarely aligned. To address the alignment issue, Uludag had originally proposed the use of a polar coordinate system where the core of the fingerprints must first be identified and relative angles of minutiae data are stored [21]. However, the use of relative angles can reveal information on the locations of the minutiae and greatly reduces the security of the system. Recently, Uludag proposed another automatic alignment method through the use of helper data. The helper data consists of maximum curvature values in the ridge pattern of the fingerprint [20]. It was claimed that the helper data does not leak any information on the minutiae location. Arathi proposed a different approach to solve the alignment problem, using rotation/translation invariant representations [2]. The author experimented with three different representations: Voronoi neighbors, five-nearest neighbour structure and triangle-based structures. The first two consider only the local structure in the matching process while the last method considers the global structure. One drawback of these approaches is that it uses vastly more information to represent the minutiae data.

The security of the Fuzzy Vault scheme is based on the polynomial reconstruction problem. Given that the attributes, i.e. the keys, are randomly selected, it can satisfy various level of security at the cost of a larger vault size (ciphertext length). When applied to biometrics, the level of security it provides becomes difficult to measure because of the fact that biometric attributes are unlikely to be random. All existing

security analysis assumes equiprobable keys.

2.1.2 Security

The security of the Fuzzy Vault scheme relies on the chaff points' ability to hide the true points. The effect of the chaff points is to increase the number of spurious polynomials. A spurious polynomial is a curve of degree k or less containing a set of exactly t points in the vault, which is not the secret polynomial. In the information theoretic sense, the security can be defined as the number of plaintexts associated with a ciphertext. For the Fuzzy Vault scheme, this translates to the number of valid messages for a particular vault. Consider a vault V_A locked under the set A, there are r sets of points, of which t are real. Then, there exists at least $\frac{\mu}{3}q^{k-t}(\frac{r}{t})^t$ spurious polynomials, with probability at least $1-\mu$, for $\mu > 0$ [8]. For example, for $\mu = 0.01$, $r = 300$, $t = 22$, $k = 19$, $q = 2^{16}$, we would have at least $108844773 \approx 2^{26.7}$ spurious polynomials, with probability 0.99 [8]. Assuming any point is equally likely to be a true point, the attacker would have to consider all these polynomials to equally likely be the secret polynomial. The system can be said to achieve 26-bit security at the 99% confidence level.

In this chapter, the Fuzzy Vault algorithms were presented. We showed that the cryptosystem works by converting a message to a set of points with redundancy, allowing errors to be corrected. To provide security, a large amount of chaff points is added to the vault to hide the true set. This leads to the definition of information theoretic security in terms of spurious polynomials. Several researchers have expressed interest in the simple scheme and provided insights on how it may be used for biometric encryption.

Chapter 3

Application to Biometrics

The Fuzzy Vault scheme can be used for authentication, where biometrics is used to encrypt a traditional cryptographic key (eg. AES). Since good cryptographic keys are often random and lengthy in nature, users cannot effectively memorize them. Instead, many systems use passwords to provide authentication. Users enter a password to release the cryptographic key. This allows for simple attacks, such as dictionary search, on the password to break the system. Using biometrics authentication has several advantages. Biometrics is difficult to copy, share and requires the person to be present to provide authentication. It doesn't have to be memorized and it uniquely identifies the user [19].

In this chapter, we explain the problems of existing biometric authentication systems and the motivation for biometric encryption. Current techniques in describing fingerprints as a set of attributes to be used in Fuzzy Vault are then presented in section 3.3 followed by some suggestions on improvements.

3.1 Conventional biometrics vs. Biometric encryption

Biometrics-based authentications can be classified into two groups: key-release and key-generation [19].

In a key-release system, the cryptographic key is stored in a secure location and released upon a correct match in the biometrics reading. Hence, the cryptosystem

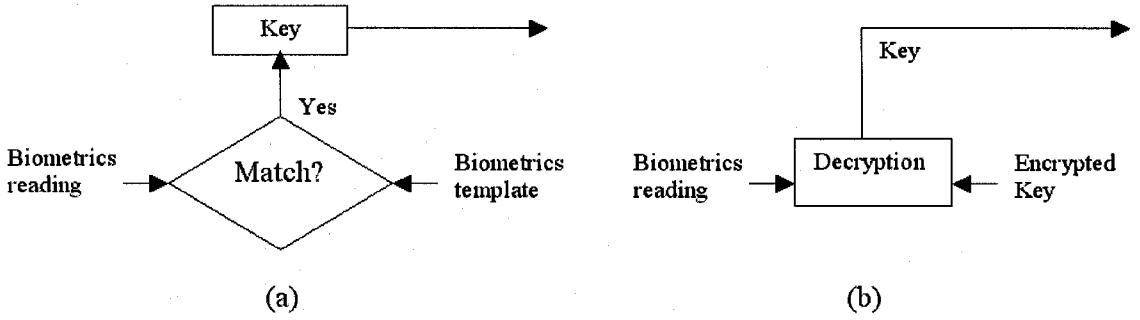

Figure 3.1: Biometrics authentication based on (a) Key release (b) Key generation

is decoupled from the biometrics matching. This allows for vulnerabilities on the matching process to compromise the system. Hill-climbing attack is often mentioned when describing vulnerabilities on key-release systems [1]. In the matching process, a match score is used to describe the likelihood that the biometrics reading corresponds to the stored template. Hill-climbing attacks make use of this score to provide adjustments by continuingly querying the system in an attempt to improve the score until the key is released. One can also attack the device to trigger a correct match response regardless of the input [1]. There is also privacy concern since the owner of the system has access to every user's biometric template. The fact that biometrics cannot be easily altered is another limitation to this biometric system. Should a user's biometrics be compromised, the security is forever lost.

In a key-generation system, the matching and the cryptosystem is merged such that a correct match produces the cryptographic key. In this system, neither the biometric template nor the secret key is available in plain form to an attacker. Since there is no match response generated, it is not possible to temper the device into releasing the key. The Fuzzy Vault is an example of a key-generation system, which is resistant towards attacks on key-release systems. If one is to consider a Hill-climbing attack on Fuzzy Vault, one would require a score based on how close a chosen set of t points is to a set of exactly t points in the vault which lie on a polynomial of degree k or less. There is no known algorithm to achieve such a score. Even if such a score can be obtained, the algorithm would not be able to differentiate the secret polynomial from the spurious polynomials. Figure 3.1 shows the comparison between the two methods.

Although the idea of the Fuzzy Vault scheme works, many issues have to be addressed before a practical system can be built, in particular in one where biometrics keys are used.

3.2 Security of biometric systems

Conventionally, security in biometric systems are measured using the genuine accept/reject rate (GAR/GRR) and the false accept/reject rate (FAR/FRR):

- Genuine accept rate (GAR): Probability that a genuine user would be accepted
- Genuine reject rate (GAR): Probability that a genuine user would be rejected
- False accept rate (FAR): Probability that an imposter would be accepted
- False reject rate (FRR): Probability that an imposter would be rejected

To evaluate these values, the system is generally tested against a large database of users.

3.3 Fingerprint Fuzzy Vault scheme

Noting that the scheme can be applied to biometrics, Clancy and Uludag independently implemented the Fuzzy Vault scheme based on fingerprints. The first task is to decide how to properly represent the biometric information as a set of values.

A fingerprint template generally consists of minutiae locations. Minutiae are places where a fingerprint ridge either splits, denoted ridge bifurcation, or ends, denoted ridge ending. It is widely known that minutiae data reliably identifies individuals and, hence, can be used as the locking set in the algorithm.

3.3.1 Minutiae representations

The choice of the minutiae representation mainly affects:

- Noise tolerance of individual minutiae
- Need for an pre-alignment algorithm

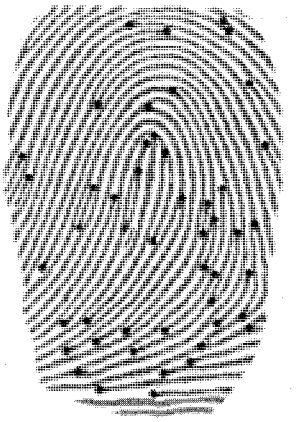

Figure 3.2: Minutiae locations highlighted on a fingerprint[18]

- Vault size
- Complexity of the encoding algorithm

In both Clancy's and Uludag's implementation [18, 19], the minutiae locations are expressed in Cartesian coordinates $P_i = (x_i, y_i)$. A locking set is produced by simply taking $a_i = x_i|y_i$, where | denotes concatenation. For example, a fingerprint extraction produces the following minutiae points: $(101, 110), (111, 010), (001, 010)$. Then, the corresponding locking set A is $(101110, 111010, 001010)$. The advantage of this method is its simplicity. However, this representation requires pre-processing to align the fingerprint.

In Uludag's implementation [19], the minutiae are first quantized to a square lattice before being used as the locking set, $P_i = Q(x_i, y_i)$. Figure 3.3 demonstrates this process. During decoding, the unlocking set is selected as all points matching those in the vault.

In Clancy's work, the coordinates (x_i, y_i) of the pixels representing the minutiae on the image are used directly as the locking set. A minimum distance is then used to ensure the points in the vault are not too close together when generating chaff points. To unlock the vault, we select the points closest to the ones in vault from the unlocking set. This is almost equivalent to quantizing the image according to a hexagonal lattice and choosing closest points during decoding. By using the coordinates

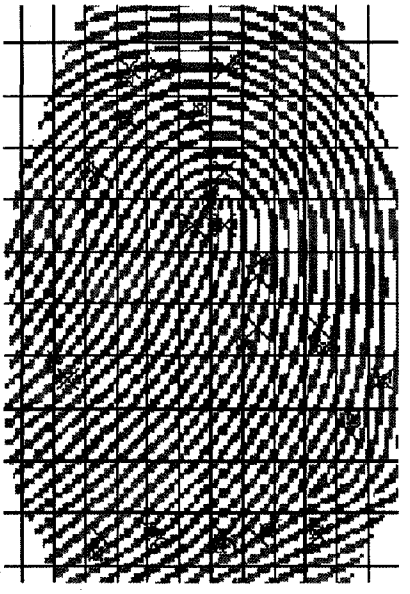

Figure 3.3: The red circles highlight the minutiae locations and the crosses denote the quantization blocks containing minutiae. The (x, y) coordinates of the blocks are then used as the locking set

directly, the symbol size and, consequently, the vault size will then be much larger than the quantized coordinates that Uludag used.

The choice of the lattice affects the noise tolerance. If the noise on the coordinates were uniformly Gaussian and of the same variance for all minutiae, then a hexagonal lattice would be optimal. However, the noise is not stationary across the fingerprint image. Due to the shape of the finger, the noise will be much smaller near the point of initial contact of the finger with the scanner and more prominent far away from the point. Given this model, a non-uniform grid representing the polar coordinate system centered on the initial contact point may be better suited to describe the varying noise levels. Since the goal of the quantization is to introduce noise tolerance, neither a square lattice nor a hexagonal lattice is optimal. Assuming the same lattice and the maximum vault size are used, the two methods of selecting from the unlocking set are almost equivalent. As the vault size decreases, the false accept rate and genuine accept rate would increase if closest points were selected as unlocking set whereas it would remain the same if matching points were used. Figure 3.4 and 3.5 illustrates the situations for maximal and non-maximal vault size. In figure 3.4 a), quantized

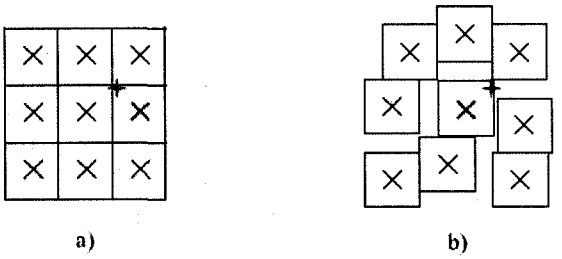

Figure 3.4: With maximal vault size, a) a quantization approach and b) a minimum distance approach to generating vault data behave similarly

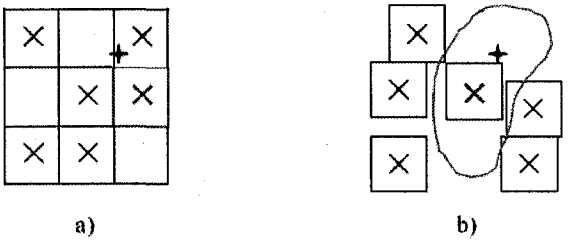

Figure 3.5: With non-maximal vault size, b) the minimum distance approach demonstrates a higher acceptance rate as seen from the expanded region which maps to the true point

values were stored as in Uludag's method. In b), the chaff points are generated by assuring that no other points are within the cell of each point, equivalent to assuring that points are at least a certain distance away in a hexagonal lattice as in Clancy's method. The red star represents an unlocking minutiae mapped to the red cross. In figure 3.5 a), the blue star, an illegitimate user's minutia, is quantized to the wrong block. However, in b), the closest point in the vault is the red cross. The area contained inside the green border represents the region where a minutia point may be mapped to the true minutia. Precisely, whether an unlocking minutiae point is mapped to the true minutiae depends on the number of points around it in the case where the closest point is selected and maximum vault size is not used. The less neighbour a true point has, the more likely an unlocking point may be matched with it. Which method to use depends on the application and how we would like to control the GAR/FAR rate.

In terms of information security (i.e. the ability to conceal the true points), Chang and Teo [3] found that Clancy's method of generating the chaff points reveals some

information on the true points. Their observation was that points that arrived earlier (true points) in the vault generation process tend to have less neighboring points compared to points that arrived later (chaff points). The observation is based on empirical results (simulation) and they were unable to prove it analytically. However, this does not mean that Clancy's method is insecure, but rather that the security the method provides is less than brute-force.

Despite its simplicity, the Cartesian coordinate representation requires the fingerprint images to be pre-aligned. Currently, there are two techniques devised to solve this problem:

1) Translation/rotation invariant representation

2) Alignment using helper data

Uludag first made use of a polar representation of the minutiae locations, using relative positioning of minutiae locations which are unaffected by translation/rotation. Its construction is discussed in details in [21]. In general, it requires the calculation of a central point where fingerprint ridges revolves around. Upon completion, the relative positions between sets of three minutiae locations are calculated and stored as a single record (x, y, q). Then, we simply take $a_i = x_i|y_i|q$ to produce the locking set A. However, it is vulnerable to erasures of minutiae points and could reduce the security due to additional information being available to the attacker.

Three other rotation/translation invariant representations were considered in [2]. In the five nearest neighbour based representation, each minutia is described by the distances from the five nearest minutiae and the angles with respect to the central minutia's ridge direction. Figure 3.6 illustrates the parameters for representing a minutia in this system. Voronoi representation is similar, as it dissects the image into Voronoi cells and represents each minutia by considering the neighbouring cells. The final representation tested was the triangle-based structures. Instead of considering only neighbours, this system describes the minutiae data by storing the relative positions of every combination of three minutia points. Although these representations are less vulnerable to erasures, they significantly increase the vault size and may further reduce the security due to information on the relative positions of minutiae being available to an attacker.

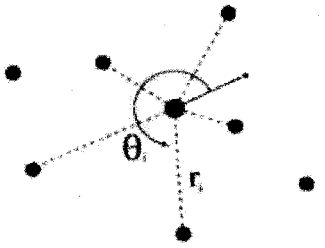

Figure 3.6: The central minutia is represented by the set $(r_1, \theta_1,\ r_2, \theta_2,\ r_3, \theta_3,\ r_4, \theta_4,\ r_5, \theta_5)$ [2]

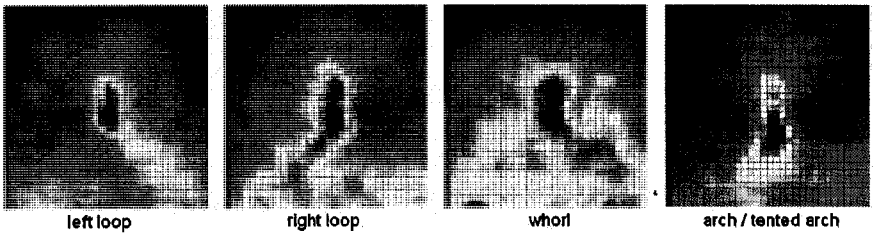

Figure 3.7: Minutiae Density on different types of fingerprints

Another technique is to include extra information in the vault for pre-alignment of the images while using the Cartesian representation. In [20], Uludag described a method of extracting the maximum curvature points of the ridge flow on the fingerprint image. These points are then used to align with the corresponding set from the unlocking fingerprint. Although the maximum ridge orientations of the fingerprint may not directly lead an attacker to the minutiae locations, we believe that it does leak some information on the minutiae locations. Since fingerprints of the same pattern (i.e. whorl, left loop, arch, etc) likely have very similar maximum curvature points, an attacker will immediately be able to identify the pattern from the helper data of the vault. Since each fingerprint pattern have a known probability distribution for minutiae, as shown in figure 3.7, the corresponding distribution can then be used to attack the vault.

Increasing the entropy of the locking set

One of the major limitations in using fingerprints is that the entropy of the templates tend to be low when compared to traditional cryptographic keys. In all existing implementations, the locking set is chosen to consist of only the minutiae locations. However, a fingerprint template often contains other information, such as the ridge direction, the minutia type, the fingerprint pattern, the core location, the local quality of the impression, etc. It may be possible to increase the entropy of the locking set by including some of these information. However, care must be taken so that the added information does not compromise the minutiae locations. Currently, the most appropriate candidate seems to be the minutia type as we believe that its correlation with the minutia locations is fairly low. If only the two most common minutiae types are used, then the entropy gained is only 1 bit. Hence, it may be interesting to also consider the less common minutiae types.

By observing the minutiae distributions in figure 3.7, we note that there is a high probability of finding a minutia near the centre of the fingerprint image. If the coordinates of the minutiae are used as the locking set, it would then always contain certain values. As a remedy to this situation, we may warp the space according to the probability distribution such that the locations with the highest probability are mapped more densely than the lower probability regions. Unfortunately, this also reduces the noise tolerance where minutiae are most likely to appear. Instead, we propose putting more weight on minutiae that appear in low probability regions. A simple method to achieve this is by simply associating more than one point to the less probable regions. For example, a minutiae appearing near the edge at $(2, 5)$ can be mapped to both $1, 3$ and $(2, 8)$. Then, it would be more valuable if the unlocking set contains this minutiae rather than one in the center.

In this chapter, we presented the advantages of the Fuzzy Vault scheme over conventional biometric systems. In particular, it is resistant to attacks on the matching process such as the Hill-climbing attack. Several researchers have proposed the use of fingerprints in Fuzzy Vault. Generally, the fingerprints are described as a set of minutiae, which is then used as the locking set. Due to variations in different impressions, the minutiae set are usually corrupted by noise. To introduce noise tolerance, different quantization techniques were proposed by Clancy and Uludag. Also, in order to use

the minutiae coordinates directly, the fingerprints must be pre-aligned. To achieve this, the addition of side-information and rotation-translation invariant representations were proposed. Unfortunately, both techniques reduce the security of the system by leaking information on the locking set. In an effort to increase the entropy of the locking set, we noted that we may add minutiae types to the locking set and place more weight on low probability minutiae.

Chapter 4

Decoding Algorithms for the Fuzzy Vault scheme

The quantization techniques discussed in the previous chapter were designed to cope with the variation in the biometric templates, specifically for fingerprints. Aside from white noise, a fingerprint template can also contain errors in the form of missing or new minutiae. Naturally, a biometric encryption scheme must then also be able to handle these errors. To this end, Juel and Sudan proposed the use of error correction code in their scheme.

The Fuzzy Vault decoding problem can be described as follows:

> *Given a set of t points, find a polynomial of degree k or less passing through most of the points.*

It turns out that this problem is intimately related to that of decoding Reed-Solomon(RS) codes. Sudan had originally suggested the Berlekamp-Massey algorithm as a potential decoder for the Fuzzy Vault scheme. However, developing such a decoder was not demonstrated. Some researchers have expressed difficulty or even questioned the validity of the use of a RS decoder [10, 19, 21]. To our knowledge, no researcher have yet to provide an implementation of a RS decoder for Fuzzy Vault. Almost all existing implementations use the CRC decoding method introduced by Uludag [19] to circumvent the RS decoding problem. The following section describes Uludag's CRC decoder for Fuzzy Vault and discusses its shortcomings, highlighting the need for a practical RS decoder. Section 4.2 provides an introduction to Reed-Solomon codes

and discusses their relationship with the Fuzzy Vault decoding problem. The chapter concludes by proposing two practical RS decoding algorithms for the Fuzzy Vault scheme.

4.1 CRC based decoding algorithm

As an alternative to a RS decoder, Uludag proposed a method to decode many candidate messages and identify the correct message through the use of Cyclic Redundancy Check (CRC).

The use of a CRC decoder requires the message to be appended with a CRC prior to the generation of the vault. During decoding, the decoder is presented with t' points. For every set of k points, it constructs a candidate message using Lagrange interpolation and attempts to verify the CRC. If the CRC is verified to be correct, the decoder outputs the candidate as the decoded message.

For example, a 8-byte message with a 1-byte CRC can be written as $\{m_0, m_1, \ldots, m_7, c_0\}$. The message polynomial is then $m(x) = m_0 + m_1 x + \cdots + m_7 x^7 + c_0 x^8$. During decoding, the decoder receives 12 points. For every set of 9 points, it constructs a candidate polynomial of degree 8: $m'(x) = m'_0 + m'_1 x + \cdots + m'_7 x^7 + c'_0 x^8$. If the candidate polynomial's CRC is verified to be correct, it outputs $\{m'_0, m'_1, \ldots, m'_7\}$ as the decoded message.

The advantage of this method is its conceptual simplicity. When used in practise, however, the decoder may have to perform hundreds or thousands of interpolations and CRC verifications before finding the correct message or declaring a failure to decode. In Uludag's initial implementation with $q = 2^{16}, r = 218, t = 18, k = 9$, a decoding operation required, on average, 52 seconds on a computer with a 3.4GHz processor. This severely limits the practicality as users would likely not want to wait for such a lengthy period of time to be authenticated. The computational load also limits the expansion to portable devices, for which applications, such as biometric smart card [18], have already been suggested. Also, one of the most interesting and desirable property of the Fuzzy Vault scheme is that it's non-binding. A single vault can be generated from many possible locking set/message pairs. Polynomials which exhibit the same behaviour as the message polynomial are called spurious polynomials. In biometric applications, having a large number of spurious polynomials allow the user's biometrics

to remain protected. Since biometrics cannot be replaced, it is of great importance that it cannot be deduced from the vault. With a large number of spurious polynomials, if an attacker gained access to a vault, he cannot, even with infinite computational power, be ascertain of the locking set or the message that was originally used to create it. By including a CRC, the message polynomial becomes structured and many of the spurious polynomials without a valid CRC become distinguishable from the message polynomial, representing a privacy concern. Interestingly, our implementations with Uludag's parameters also showed that, when a large number of chaff points is used to increase security, the increased number of interpolations required can result in over 50% chance of CRC collision. Even when a legitimate unlocking set is presented, there is a high probability that the incorrect message is decoded.

Due to these disadvantages, a CRC decoder is not suitable in practise. The scheme requires an efficient decoding algorithm, which continues to allow the locking set and message to remain well hidden. To achieve this goal, we return to Sudan's view of the unlocking point set as a Reed-Solomon code word.

4.2 Reed-Solomon Codes

Reed-Solomon (RS) codes are arguably one of the most powerful error correction codes discovered in the twentieth century. Due to their burst error correction capability, RS codes have found many applications in storage systems, such as compact discs where scratches are common. When combined with convolutional codes, RS codes had even played a central role in various NASA missions, enabling reliable data transmission through billions of kilometres from the outer regions of our solar system.

Irving Reed and Gustave Solomon first introduced RS codes in a paper entitled "Polynomial Codes over Certain Finite Fields" [14] in 1960. Being a maximum distance separable code, a (n, k) RS code can correct up to $t = \lfloor \frac{(n-k+1)}{2} \rfloor$ errors, which is the best possible error correction capability for any code of same length and dimension [22].

4.2.1 The original approach to RS codes

The original construction of RS codes is extremely simple. Given k message symbols $M = \{m_i\}_{i=0}^{k-1}$ where $m_i \in \mathrm{GF}(q)$, a message polynomial $m(x) = m_0 + m_1 x + m_2 x^2 + \ldots + m_{k-1} x^{k-1}$ can be constructed. The RS code word corresponding to M is computed as [22]

$$C = \{c_0, c_1, c_2, \ldots, c_{q-2}, c_{q-1}\} = \{m(0), m(a), m(a^2), \ldots, m(a^{q-2}), m(1)\} \qquad (4.1)$$

where a is a generator of $\mathrm{GF}(q)$. In another words, it is simply the evaluation of the message polynomial at q points. The length of this RS code, n, is q. Since $m(x)$ has degree $k-1$, it can be determined from any k points. Suppose that the received code word contains t errors, then $q-t$ symbols remain correct and lie on $m(x)$. There can be at most $t+k-1$ points lying on a polynomial other than $P(x)$. Therefore, it is possible to recover the message given that $q-t > t+k-1$ or, equivalently, $t < \frac{(n-k+1)}{2}$.

4.2.2 The generator polynomial approach to RS codes

Soon after Reed and Solomon's paper was published, Gorenstein and Zierler [6] noticed that RS codes are equivalent to a special case of BCH codes and can be encoded via generator polynomials. At the time, it was felt that the generator polynomial approach would lead to more efficient decoding algorithms. Hence, the original approach was mostly abandonned and the new approach became the standard method for constructing RS codes.

The first step to constructing a t-error correcting RS code using this method is to choose a proper generator polynomial, which contains $2t$ consecutive powers of a as roots:

$$g(x) = \prod_{i=b}^{b+2t-1} (x - a^i) \qquad (4.2)$$

where b is any integer less than q. The length of this RS code, n, is $q-1$. Similar to the original method, a message consisting of k symbols $M = \{m_i\}_{i=0}^{k-1}$ is encoded as a polynomial $m(x) = m_0 + m_1 x + m_2 x^2 + \ldots + m_{k-1} x^{k-1}$. The corresponding RS code

word is simply the polynomial multiplication of the message with the generator:

$$c(x) = m(x)g(x) \tag{4.3}$$

The code word polynomial can be written as $c(x) = c_0 + c_1x + c_2x^2 + \ldots + c_{q-2}x^{q-2}$. Since the generator has $2t$ consecutive roots, the code word must also contain the same $2t$ roots. This leads to a very simple method for detecting errors. Since the degree of $m(x)$ is $k-1$ and the degree of $g(x)$ is $2t$, we have $q = k + 2t + 1$ or, equivalently, $t = \frac{n-k}{2}$.

4.2.3 Equivalence of the two approaches

At first glance, the two approaches seem very different. However, we can verify that, by reducing the original code by one symbol, they are indeed equivalent in the sense that they produce the same set of code words from the same message space.

Consider a RS code word generated using the original approach with a slight rearrangement of the symbols:

$$C = \{c_0, c_1, \ldots, c_{q-2}\} = \{m(1), m(a), \ldots, m(a^{q-2})\} \tag{4.4}$$

In the generator polynomial approach, the code word polynomial is defined as $c(x) = \sum_{l=0}^{q-2} c_l x^l = c_0 + c_1x + c_2x^2 + \ldots + c_{q-2}x^{q-2}$. Note that $c_l = m(a^l) = \sum_{j=0}^{k-1} m_j a^{jl}$. Then, we can rewrite:

$$c(x) = \sum_{l=0}^{q-2} \sum_{j=0}^{k-1} m_j a^{jl} x^l \tag{4.5}$$

Evaluating at a^i gives:

$$\begin{aligned} c(a^i) &= \sum_{l=0}^{q-2}\sum_{j=0}^{k-1} m_j a^{jl} a^{il} \\ &= \sum_{l=0}^{q-2}\sum_{j=0}^{k-1} m_j a^{(j+i)l} \\ &= \sum_{j=0}^{k-1} m_j \sum_{l=0}^{q-2} a^{(j+i)l} \end{aligned} \tag{4.6}$$

From theorem B.1.1,

$$\sum_{l=0}^{q-2} a^{(j+i)l} = 0 \quad \text{if} \quad j+i \neq 0 \mod q-1 \tag{4.7}$$

Since $0 \leq j \leq k-1$ and $q = k+2t+1$, then the condition holds for $0 < i \leq 2t$:

$$c(a^i) = \sum_{j=0}^{k-1} m_j \sum_{l=0}^{q-2} a^{(j+i)l} = 0 \quad \text{for} \quad 0 < i \leq 2t \tag{4.8}$$

Thus, a RS code word generated from equation 4.4 has $2t$ consecutive roots. This means it is a valid code word as defined using the generator polynomial approach. Simple arguments in appendix B.2 can be used to show that different messages would lead to different code words and that there are exactly q^k valid code words for both approaches. Therefore, any valid code word constructed using the original approach is also a valid code word using the generator approach, and vice-versa. The sole difference is that the same message is mapped to different code words when using different approaches.

4.2.4 RS codes in Fuzzy Vault

Recall from section 2.1, a Fuzzy Vault is generated by first considering the message as a polynomial and evaluating it at t locations from the locking set. This process is essentially the original approach to constructing RS codes, except the message polynomial is evaluated at only t points instead of $q-1$. The resulting set of points is called the true point set and is combined with a random set of chaff points to produce the vault.

The set of chaff points can be thought of as errors. During decoding, the Fuzzy Vault decoder is presented with the unlocking set, $B = \{b_i\}_{i=1}^{t'}$ and the corresponding set of points, $Q = \{q_0, q_1, \ldots, q_l\} = \{(x_i, y_i)\}_{i=1}^{l}$ where $x_i \in B$, from the vault is extracted. Q may contain some true points and some chaff points. Decoding is successful if there is significantly more true points than chaff points.

The set Q can be thought of as a RS code word constructed using the original approach, with missing elements denoted by the erasure symbol, E:

$$Q_{RS} = \{E, E, \ldots, E, y_0, E, E, \ldots, E, y_1, E, E, \ldots\ldots, E, y_l, E, E, \ldots, E\} \tag{4.9}$$

where the x_i^{th} symbol in the code word is y_i. Erasures are symbols that are missing or unreadable. Unlike errors, erasure locations are known and this information allows the decoder to be better equiped when dealing with erasures. In fact, RS decoders can correct twice as many erasures as errors. Let e denote the number of erasures and v denote the number of errors, the message can be recovered if $v + \frac{e}{2} < \frac{(n-k+1)}{2}$.

For the Fuzzy Vault scheme to be secure, the number of chaff points must be much greater than the number of true points. The unlocking set and, consequently, Q, will correspond to a small subset of the vault. Hence, the RS codeword derived from Q will contain an unproportionally large number of erasures, leading to some practical considerations discussed in the following section.

4.3 RS decoding algorithms for the Fuzzy Vault scheme

The analysis from the previous section showed that a Fuzzy Vault decoder must be able to recover the message from a RS codeword constructed using the original approach with erased and erroneous symbols.

4.3.1 The Berlekamp-Massey algorithm

The Berlekamp-Massey algorithm was originally suggested by Sudan as a RS decoder for Fuzzy Vault. However, the algorithm has always been described from the viewpoint of the generator polynomial approach. The equivalence demonstrated in section 4.2.3

is only in the sense that the two approaches provide a mapping from the same message space to the same code word space. The mapping itself is not identical and the two approaches do not map the same message to the same code word. In another words, a RS decoder for the generator polynomial approach will not correctly recover the message from a RS code word generated using the original approach. Despite this difference, a RS decoder using the Berlekamp-Massey algorithm can be designed to decode RS code words from the original approach. To better understand how this is achieved, we'll begin with the description of the Berlekamp-Massey algorithm from the generator viewpoint [15].

A received RS code word with some erroneous symbols can be represented as follows:

$$r(x) = c(x) + e(x) = g(x)m(x) + e(x) \tag{4.10}$$

where $e(x)$ represents the error pattern. Suppose the received code word contains v errors, then $e(x)$ can be expressed as:

$$e(x) = e_{j_1}x^{j_1} + e_{j_2}x^{j_2} + \ldots + e_{j_v}x^{j_v} \tag{4.11}$$

where e_{j_i} are the error values and j_i are the error locations. The first step in the decoding algorithm is to determine the syndromes. Syndromes are defined as:

$$S_i = r(a^i) = g(a^i)m(a^i) + e(a^i) = e(a^i) \tag{4.12}$$

or, substituting from equation 4.11,

$$S_i = e_{j_1}a^{ij_1} + e_{j_2}a^{ij_2} + \ldots + e_{j_v}a^{ij_v} \tag{4.13}$$

We then define the error-location polynomial as

$$\begin{aligned} \sigma(x) &= (1 - a^{j_1}x)(1 - a^{j_2}x)\ldots(1 - a^{j_v}x) \\ &= \sigma_0 + \sigma_1 x + \ldots + \sigma_v x^v \end{aligned} \tag{4.14}$$

The coefficients of $\sigma(x)$ and the syndromes are related by the following sets of

Table 4.1: Berlekamp-Massey algorithm for finding $\sigma(x)$

μ	$\sigma^{(\mu)}(x)$	d_μ	l_μ	$\mu - l_\mu$
-1	1	1	0	-1
0	1	S_1	0	0
1	$1 - S_1 x$			
2				
$\vdots$				
$2t$				

equations, known as the generalized Newton's identities:

$$\begin{aligned} S_{v+1} + \sigma_1 S_v + \sigma_2 S_{v-1} + \ldots + \sigma_v S_1 &= 0 \\ S_{v+2} + \sigma_1 S_{v+1} + \sigma_2 S_v + \ldots + \sigma_v S_2 &= 0 \\ &\vdots \\ S_{2t} + \sigma_1 S_{2t-1} + \sigma_2 S_{2t-2} + \ldots + \sigma_v S_{2t-v} &= 0 \end{aligned} \tag{4.15}$$

The goal of the Berlekamp-Massey algorithm is to determine an error-location polynomial of the smallest degree which satisfies these identities. Note that once $\sigma(x)$ is found, the error locations are simply the reciprocal of its roots.

Given the syndromes, $\{S_i\}_{i=1}^{2t}$, the iterative Berlekamp-Massey algorithm proceeds as follows:

1) Initialize the variables as indicated in table 4.1, where d_μ represents the discrepancy at the μ^{th} step of the algorithm and l_μ is the degree of $\sigma^\mu(x)$

2) At the μ^{th} step, compute the discrepancy $d_\mu = S_{\mu+1} + \sigma_1^{(\mu)} S_\mu + \ldots + \sigma_{l_\mu}^{(\mu)} S_{\mu+1-l_\mu}$, where $\sigma_i^{(\mu)}$ are the coefficients of $\sigma^{(\mu)}(x)$

3) If $d_\mu = 0$, $\sigma^{\mu+1}(x) = \sigma^\mu(x)$.

4) If $d_\mu \neq 0$, $\sigma^{\mu+1}(x) = \sigma^\mu(x) + d_\mu d_p^{-1} x^{\mu-p} \sigma^{(p)}(x)$, where p is chosen such that $p < \mu$, $d_p \neq 0$ and $p - l_p$ is maximal.

If less than t errors occurred, $\sigma^{2t}(x)$ is the true error location polynomial:

$$\sigma(x) = \sigma^{2t}(x) \tag{4.16}$$

The roots of $\sigma(x)$ can be found by evaluating it at every elements of GF(q). If $\sigma(a^i) = 0$, then a^i is a root and $a^{-i} = a^{q-1-i}$ is the error location.

Once the error locations are found, the error values can be obtained by first computing

$$\begin{aligned} Z_o(x) &= S_1 + (S_2 + \sigma_1 S_1)x + (S_3 + \sigma_1 S_2 + \sigma_2 S_1)x^2 \\ &\quad + \ldots + (S_v + \sigma_1 S_{v-1} + \ldots + \sigma_{v-1} S_1)x^{v-1} \end{aligned} \tag{4.17}$$

Then, the error value, e_{j_i}, at location a^{j_i} is

$$e_{j_i} = \frac{-Z_o(a^{-j_i})}{\sigma'(a^{-j_i})} \tag{4.18}$$

where $\sigma'(x)$ is the derivative of the error location polynomial, $\sigma(x)$.

Example 4.3.1. *Consider the (7,3) 2-error correcting RS code with* $g(x) = (x+a)(x+a^2)(x+a^3)(x+a^4)$, *where the field elements from* GF(2^3) *are generated by the irreducible polynomial* $f(x) = x^3 + x + 1$, *i.e.* $a^3 = a + 1$.

Let the transmitted code word be $c(x) = 1 + a^3x + ax^2 + ax^3 + x^4 + a^3x^6$ *and the received word be* $r(x) = a^3x + ax^2 + ax^3 + x^4$. *Upon receiving the code word, the decoder first computes the syndromes:*

$$\begin{aligned} S_1 &= r(a) &&= a^4 + a^3 + a^4 + a^4 &&= a^6 \\ S_2 &= r(a^2) &&= a^5 + a^5 + a^7 + a^8 &&= a^3 \\ S_3 &= r(a^3) &&= a^6 + a^7 + a^{10} + a^{12} &&= 0 \\ S_4 &= r(a^4) &&= a^7 + a^9 + a^{13} + a^{16} &&= a^2 \end{aligned}$$

Then, the error location polynomial is found using the Berlekamp-Massey algorithm, as shown in table 4.2.

Since $t = 2$, *the algorithm stops at the* $2t = 4^{th}$ *step. We have* $\sigma(x) = 1 + a^2x + a^6x^2 = (1 + x)(1 + a^6x)$, *with* 1 *and* a^{-6} *being the roots. Taking the inverse, the error locations are found to be* 1 *and* a^6, *as expected.*

Table 4.2: Berlekamp-Massey algorithm for finding $\sigma(x)$ in example 4.3.1

μ	$\sigma^{(\mu)}(x)$	d_μ	l_μ	$\mu - l_\mu$
-1	1	1	0	-1
0	1	a^6	0	0
1	$1 + a^6x$	a^2	1	0 (p=0)
2	$1 + a^4x$	1	1	1 (p=0)
3	$1 + a^4x + ax^2$	a	2	1 (p=2)
4	$1 + a^2x + a^6x^2$			

To determine the error values, we compute

$$\begin{aligned} Z_o(x) &= S_1 + (S_2 + \sigma_1 S_1)x + (S_3 + \sigma_1 S_2 + \sigma_2 S_1)x^2 \\ &= a^6 + (a^3 + a^2 a^6)x + (0 + a^2 a^3 + a^6 a^6)x^2 \\ &= a^6 + (a^3 + a^8)x + (a^5 + a^{12})x^2 \\ &= a^6 + x \end{aligned}$$

and

$$\begin{aligned} e_1 &= \frac{-Z_o(1)}{\sigma'(1)} = \frac{a^6 + 1}{a^2} = 1 \\ e_{a^6} &= \frac{-Z_o(a^{-6})}{\sigma'(a^{-6})} = \frac{a^6 + a}{a^2} = a^3 \end{aligned}$$

Thus,

$$\begin{aligned} e(x) &= 1 + a^3x^6 \\ c(x) &= r(x) - e(x) = 1 + a^3x + ax^2 + ax^3 + x^4 + a^3x^6 \end{aligned}$$

Finally, the message is obtained by division:

$$m(x) = \frac{c(x)}{g(x)} = a^3x^2 + a^6x + a^4$$

If the received code word contains e erased symbols in addition to v symbol errors, the message can also be decoded if $v + \frac{e}{2} < \frac{(n-k+1)}{2}$. To correct erasures, we must first construct an erasure polynomial in a similar fashion to $\sigma(x)$:

$$B(x) = (1 - a^{j_1}x)(1 - a^{j_2}x)\ldots(1 - a^{j_e}x) \tag{4.19}$$

where j_i are the erased locations. The erased symbols are then filled with zeros and the syndromes are computed. The syndrome polynomial is defined as:

$$S(x) = S_1 + S_2x + \ldots + S_{2t}x^{2t-1} \tag{4.20}$$

To account for the known erased locations, the syndromes are then modified by multiplying $S(x)$ with the erasure polynomial and taking the first $2t$ coefficients:

$$S'(x) = [S(x)B(x)]_{2t} = S'_1 + S'_2x + \ldots + S'_{2t}x^{2t-1} \tag{4.21}$$

The modified syndromes are then used in the Berlekamp-Massey algorithm to determine the error locations in the unerased portion of the code word.

Berlekamp-Massey decoder for the original approach to RS codes

Our goal is to employ the Berlekamp-Massey algorithm described in the previous section to decode RS codes constructed from the original approach. It is interesting to note that the Berlekamp-Massey algorithm can correct t errors in a n-symbol code word as long as it contains $2t$ known consecutive roots and the message contains k symbols, where $t = \frac{n-k}{2}$. Knowledge of the mapping of the message to the codeword is not necessary to perform the correction. The only information required by the algorithm are $2t$ syndromes, which are computed by evaluating the received word at the $2t$ known roots. It was shown in section 4.2.3 that a RS code word generated from the original approach does contain these roots. Hence, it is possible to use the Berlekamp-Massey algorithm to correct the errors in such a RS code word. However, some improvement can be made when adapting the algorithm to the original approach.

The simplest method to recover the message from a RS code word in the original scheme is by Lagrange interpolation. A message polynomial of degree $k - 1$ can be determined given k points which lie on it. This corresponds to k symbols in a code

word of length $q-1$:

$$C = \{m(1), m(a), m(a^2), \ldots, m(a^{q-2})\} \tag{4.22}$$

In another words, the message can be determined from any k correct symbols in the code word. It is not necessary to correct the entire codeword to determine the message, as in the generator polynomial approach where the message is obtained by dividing the codeword by the generator. In fact, the ability to correct symbol errors is not needed in a RS decoder for the original approach. Recall the a RS code word is successfully decoded if $2v+e < n-k+1$, the condition can be rewritten as $n-(2v+e) > k-1$, where $n-(2v+e)$ is the number of correct symbols. Hence, if the message is recoverable, the code word must contain at least k correct symbols. When using the Berlekamp-Massey algorithm in a RS decoder for the original approach, it is therefore only necessary to determine the error location and perform interpolation on any k correct symbols. The work of determining the error values and a polynomial division is replaced with that of the Lagrange interpolation.

The advantage of the Berlekamp-Massey algorithm is that it is well studied and can be implemented efficiently [8]. When used for Fuzzy Vault, the task consists of finding the polynomial on which most of the points in $Q = \{(x_i, y_i)\}_{i=1}^{l}$ lies. This can be achieved by decoding Q_{RS} defined in equation 4.9. However, unlike a transmitted code word in communication systems, Q_{RS} is composed mainly of erased symbols. The result is a potentially significant amount of computations required when constructing the erasure polynomials to modify the syndromes.

We consider several methods to efficiently compute the modified syndromes, S_i'. The equations relating the erasure polynomial and the modified syndromes are:

$$\begin{aligned} B(x) &= \prod_{i=1}^{e}(1 - a^{j_i}x) \\ S(x) &= S_1 + S_2x + \ldots + S_{2t}x^{2t-1} \\ S'(x) &= [S(x)B(x)]_{2t} \end{aligned} \tag{4.23}$$

Since only the first $2t$ coefficients are retained from the polynomial multiplication,

$$S'(x) = S(x)[B(x)]_{2t} \tag{4.24}$$

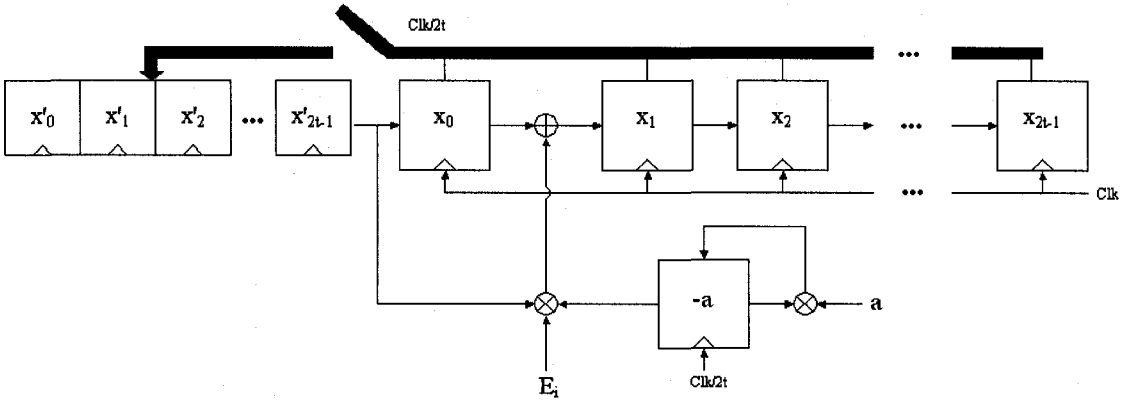

Figure 4.1: A modified syndrome polynomial evaluator circuit

A straightforward approach would then be to directly compute $B(x)$ and ignore any terms with degree greater than $2t-1$. However, this would not be very efficient. Instead of expanding $B(x)$, we can perform the multiplication of $S(x)$ with a factor of $B(x)$ at each iteration:

$$S'(x) = S(x)(1-a^{j_i}x) = [S(x) - a^{j_i}xS(x)]_{2t} \tag{4.25}$$

This operation can be implemented efficiently using shift registers. Figure 4.1 shows a circuit which computes $S'(x)$, where $E_i = 1$ if a_i is an erased symbol and Clk/$2t$ triggers every $2t$ Clk cycles. The input registers, x'_i, is initialized to $S(x)$. Each polynomial multiplication takes $2t$ clock cycles to complete and the result is fed back as the input to be multiplied by the next term. Once all the erased locations are accounted for, the x_i registers contain the modified syndromes, $S'(x)$. Note that the multiplication and addition shown are field operations.

Instead of using the erased locations, one can also compute the erasure polynomial by considering the available symbol locations by making use of the following identity:

$$\begin{aligned}\prod_{i=1}^{q-1}(1-a^ix) &= (\prod_{i=1}^{q-1}-a^i)\prod_{i=1}^{q-1}(x-a^{-i}) = (\prod_{i=1}^{q-1}-a^i)(x^n-1) \\ &= (-1)^{q-1}a^{q/2(q-1)}(x^n-1) = (-1)^{q-1}(x^n-1)\end{aligned} \tag{4.26}$$

Note that

$$\prod_{i=1}^{q-1}(1-a^i x) = \prod_{i=1}^{q-1-e}(1-a^{k_i}x)\prod_{i=1}^{e}(1-a^{j_i}x) \tag{4.27}$$

where $\{k_1, \cdots, k_{q-1-e}\}$ is the complement of the set $\{j_1, \cdots, j_e\}$ in $\{1, \cdots, q-1\}$. Hence,

$$B(x) = \prod_{i=1}^{e}(1-a^{j_i}x) = \frac{(-1)^{q-1}(x^n-1)}{\prod_{i=1}^{q-1-e}(1-a^{k_i}x)} \tag{4.28}$$

Let $q = 2^m$ and l denote the number of available symbols in Q_{RS}, then

$$B(x) = \frac{1+x^n}{\prod_{i=1}^{l}(1+a^{k_i}x)} \tag{4.29}$$

Since $l \ll e$, it is likely much easier to compute the product of the l terms in the denominator, although the division by a polynomial of degree n is a costly operation. With this formula, $e-l$ polynomial multiplications are replaced with a division. Unlike the previous approaches, it is also necessary to remember all the terms at every step until $B(x)$ is found.

Finally, we can also make use of transform techniques. The Galois field Fourier transform (GFFT) of a polynomial $v(x) = \prod_{i=0}^{q-2} v_i x^i$ over GF(q) is defined as:

$$V(x) = \prod_{j=0}^{q-2} V_j x^j \tag{4.30}$$

where

$$V_j = v(a^j) = \sum_{i=0}^{q-2} v_i a^{ij} \tag{4.31}$$

The inverse transform is performed as follows:

$$v_i = \frac{1}{n \bmod p} V(a^{-i}) = \frac{1}{n \bmod p}\sum_{j=0}^{q-2} V_j a^{-ij} \tag{4.32}$$

An immediate result from these definitions is that $V_j = 0$ if a^j is a root of $v(x)$ and $v_i = 0$ if a^{-i} is a root of $V(x)$. If we take the GFFT of $B(x)$, the result must contain e zeroes at the locations $\{V_{-j_i}\}_{i=1}^{e}$ since $\{B(a^{-j_i}) = 0\}_{i=1}^{e}$. Similar to the discrete Fourier transform, a (cyclic) convolution of the time domain symbols is reflected by a

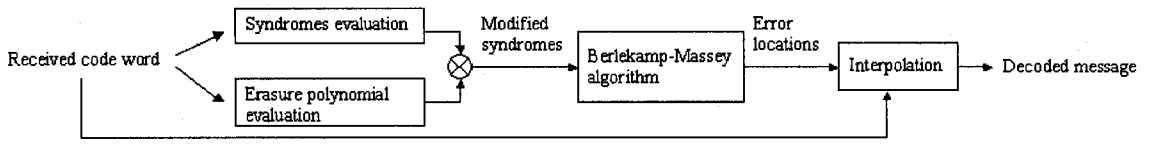

Figure 4.2: Berlekamp-Massey decoder for the original approach to RS codes

term by term multiplications in the frequency domain. A polynomial multiplications is simply a convolution of the coefficients. Thus, to obtain the GFFT of $B(x)$, we simply need to compute the l spectral components of $\{(1 - a_{j_i}x^i)\}_{i=1}^{e}$ for which the GFFT of $B(x)$ would be non-zero and perform term-wise multiplications. $B(x)$ can then be obtained through the inverse transform. The motivation for this method is that term-wise multiplications is less costly than convolutions. Since e is large, the saving may outweight the cost of performing the GFFT.

Due to time constraint, the complexity of these methods for computing the erasure polynomial have not been carefully analyzed.

A general description of a RS decoder for the original approach using Berlekamp-Massey algorithm is depicted in figure 4.2.

4.3.2 The Euclidean algorithm

Another commonly used algorithm for decoding RS codes is the Euclidean algorithm. We can easily adapt the Euclidean decoding algorithm to the original approach with the understanding from the previous section. Instead, we concentrate on a Euclidean decoding algorithm by Gao [5] designed specifically for the original approach. The algorithm does not make use of syndromes and the message is directly decoded from the code word.

Suppose a code word of length n for a message, $M = m_0 + m_1x + \ldots + m_{k-1}x^{k-1}$, is generated by

$$C = \{c_0, c_1, \ldots, c_n\} = \{m(a_0), m(a_1), \ldots, m(a_n)\} \tag{4.33}$$

where $n > k$ and a_i is the encoding set consisting of any n elements of GF(q). We

define

$$g_0 = \prod_{i=0}^{n-1}(x - a_i) \tag{4.34}$$

Given a potentially corrupted received word $r = \{r_0, r_1, \cdots, r_{n-1}\}$, Gao's algorithm proceeds as follows [5]:

1) Find a polynomial, $g(x)$, of degree $n-1$ or less such that

$$g_1(a_i) = r_i, 0 \leq i \leq n-1 \tag{4.35}$$

2) Using the extended Euclidean algorithm on g_0 and g_1, find $g(x)$ such that

$$\begin{gathered} u(x)g_0(x) + v(x)g_1(x) = g(x) \\ \deg g(x) < \tfrac{1}{2}(n+k) \\ \text{maximize } \deg g(x) \end{gathered} \tag{4.36}$$

3) Divide $g(x)$ by $v(x)$ to obtain the following equation:

$$g(x) = f_1(x)v(x) + \delta(x) \leq i \leq n-1 \tag{4.37}$$

If $\delta(x) = 0$ and $deg(f_1(x)) < k$, select $f_1(x)$ as the decoded message. Otherwise, the decoding operation fails.

The extended Euclidean algorithm is the key to the decoding algorithm. In fact, $v(x)$ is the error locator polynomial where $V(a_i) = 0$ points to a symbol error in r_i. To clarify, the goal is to solve the following equivalence for the conditions given in step 2 [4]:

$$v(x)g_1(x) = g(x) \mod g_0(x) \tag{4.38}$$

If decoding succeeds, $v(a_i) = 0$ implies $g(a_i) = 0$ and $g_1(x)$ is obtained by division.

The main advantage of Gao's algorithm is its conceptual simplicity. Despite its simple description, it has similar complexity to the Berlekamp-Massey algorithm [5]. Unlike the former, Gao's scheme allows arbitrary encoding set, averting the problem of having to compensate for the large number of erased symbols. For this reason, Gao's decoding algorithm may be more efficient as a RS decoder for Fuzzy Vault. The

decoding method can also employ FFT techniques to produce efficient implementations, in particular Cantor's algorithms for GF(2^m) [5].

4.3.3 Implementation of a RS decoder for Fuzzy Vault

To compare their efficiencies, we implemented both RS decoding algorithms for Fuzzy Vault in C using GF(2^{16}), $k = 8$.

The implementation of Gao's RS decoder for Fuzzy Vault was able to perform a complete unlocking operation in under 1 ms on a computer with a Pentium 4, 2.6GHz processor and 1GB ram, which represents a significant improvement over CRC decoders. The decoding speed was consistent regardless of failure or success in retrieving the message. The error correction capability is also faithful to the RS decoding limit: $t < \frac{(n-k+1)}{2}$.

The implementation of Berlekamp-Massey decoder for Fuzzy Vault highlighted the slowdown caused by the erasures and a sharp decrease in performance when using fields of higher order. Given that $n = q-1$ in the generator polynomial approach to RS codes, we have $2t = n - k = 65528$ syndromes. Despite being simple polynomial evaluations, the syndrome calculation already required 31 ms. When using the approach from equation 4.25, the erasure polynomial evaluation required several minutes to complete.

Due to the inflexibility of the Berlekamp-Massey algorithm with respect to codeword size and the significant work needed to process a RS codeword with large amount of erasures, it does not seem to be a good candidate as a decoder for Fuzzy Vault. On the other hand, Gao's RS decoding algorithm appears to perform well as a decoder for Fuzzy Vault due to its consideration of the original approach to RS codes and its ease in handling erasures.

4.3.4 On computational complexity and security

Since the publication of Sudan's work, it became evident that there are two different ways to evaluate the security of the Fuzzy Vault scheme. The original paper used the information theoretic security, defined in terms of the number of spurious polynomials, as an indicator of the security provided by the scheme. Various researchers later measured the security as the expected number of trials required to obtain the message in a brute force attack. We'll denote this as the computational security. In fact, both

definitions are important in characterizing the security and privacy provided by the scheme.

The information theoretic security is defined as the number of plaintexts associated with a particular ciphertext. In Fuzzy Vault, this is the number of messages which could have generated a particular vault. In Clancy's smart card construction, this roughly translates to the number of fingerprints that could be associated to the owner of a card. This is one of the important benefits of the Fuzzy Vault scheme: the plaintext is not binded to the ciphertext, unlike in some cryptographic schemes like RSA. With suitable parameters, the number of false fingerprints can be made large enough that the user's true fingerprint remains well hidden. The importance of this property should be highlighted in biometric applications. Since biometrics cannot be replaced, if it's possible to obtain a user's biometrics, it could be worthwhile for an attacker to perform a brute force attack even at the cost of years of computations. Therefore, we may say that the information theoretic security provides a measure of the level of privacy, by rendering the user's information indistinguishable from potentially thousands of other users.

The computational security usually refers to the amount of computation required to obtain the plaintext using the most efficient algorithm available, assuming that the correct plaintext can be verified. Most public key cryptosystems rely on computational security to establish secure communications. For example, RSA's security is based on the problem of factoring large numbers. If one can factor the modulus n, one can obtain the private key and perform decryption. However, if n is sufficiently large, the problem is considered impractical and can take hundreds and thousands of years to perform with the most powerful computer today. In the case of Fuzzy Vault, the computational security is usually expressed as the expected amount of computation before stumbling on the correct plaintext and it will likely be less than that of RSA, especially in biometric applications. However, it should still represent a significant amount of computation that an attacker must undertake to retrieve a set of candidates.

Several researchers [18, 19] have evaluated the computational security of Fuzzy Vault by assuming a brute force search. Let an attacker be given a vault with r sets of points, of which t are true points and lie on a polynomial of degree $k-1$. A brute force search would consist of performing Newtonian interpolation[18] on sets of k points until the secret message polynomial is found. Newtonian interpolation is

simply a reformulation of the interpolation polynomial for efficient computations in applications where points are frequently added/removed. Since there are a total of $\binom{r}{k}$ sets of k points where $\binom{t}{k}$ sets interpolate to the secret polynomial, the expected number of trials to obtain the message is[19]:

$$C_{bf} = \binom{r}{k}\binom{t}{k}^{-1} \tag{4.39}$$

Furthermore, Clancy[18] noted that, for "reasonable" parameters, there exists a δ such that the expected number of polynomials of degree $k-1$ is less than 1 and $k < \delta < t$:

$$\delta \geq \left\lceil \frac{\log \frac{1}{3}p^{2k}}{\log \frac{kp^2}{r}} \right\rceil \tag{4.40}$$

Then, by interpolating δ points at a time, we would arrive at a unique polynomial: the secret polynomial. However, in the example presented in the paper, the information theoretic security is 0. Hence, such a δ exists should come as no surprise. In addition, by considering sets of δ points, the number of polynomials of degree k being greater than 1 is just as likely as being less than 1. The expected value does not reveal information on the probability distribution. It should also be noted that the degree of the polynomial can easily be increased by adding random coefficients, subsequently removed during unlocking. By increasing k, the value computed for δ can be such that it is greater than t. It is interesting to note that, for the parameters given in the example, the information theoretic security also increases as k increases. Clancy's analysis raised an interesting point in the security evaluation: we need not be restricted to consider sets of k points to determine a degree $k-1$ polynomial when $t > k$.

One of the main advantage of considering sets of $\delta > k$ points is that any resulting polynomial of degree greater than $k-1$ can be immediately rejected. In contrast, interpolating sets of k points will always result in a polynomial of degree $k-1$ or less, requiring verification against the rest of the vault elements to confirm its candidacy. On the other hand, considering larger sets also leads to having to consider more sets. Since $r \gg t$, $\binom{r}{\delta}\binom{t}{\delta}^{-1} > \binom{r}{k}\binom{t}{k}^{-1}$. Heuristic testing of various parameters seem to indicate that, despite the costly verification step, it may be best to simply consider sets of k points, as a single verification step is likely faster than dozens of interpolations.

The previous analysis assumes that the best algorithm to discover the secret message

from the vault is by interpolating sets of points. However, the existence of a decoding algorithm could provide a more efficient method to parse through the data. In fact, the RS decoding algorithms presented in this chapter allow us to recover a message despite the presence of errors in the set of points being considered. This leads to the possibility of larger sets that include true and chaff points decoding successfully to the secret polynomial. Let's consider the use of RS code words of length n. A code word is decoded successfully if at least $\frac{n+\delta}{2}$ symbols lie on a polynomial of degree δ or less. Similar to the previous case, there are $\binom{r}{n}$ sets of n points in the vault. The code words that decode successfully are ones which contain $\frac{n+\delta}{2}$ or more points in the true set and the remaining points in the chaff set. Therefore, there are $\sum_{i=\frac{n+\delta}{2}}^{n} \binom{r-t}{n-i}\binom{t}{i}$ number of sets that decode successfully. Since we can have at most t true points, $\frac{n+\delta}{2} \leq t$. The code word size is then upper bounded by $2t - \delta$. With additional constraint to assure that no more true or chaff points are selected than the ones available, we obtain for the complexity of using RS decoding for Fuzzy Vault [18]:

$$C_{rs} = \binom{r}{n}\left(\sum_{i=\max(\frac{n+\delta}{2},n-r+t)}^{\min(n,t)} \binom{r-t}{n-i}\binom{t}{i}\right)^{-1} \tag{4.41}$$

where $\delta \leq n \leq \min(r, 2t - \delta)$. Clancy's analysis assumes that a vault would contain a unique polynomial of degree k passing through δ points. However, since such a situation cannot be guaranteed and we found that the complexity is always higher when choosing $\delta > k$, it may be better to consider $\delta = k$, a worst case scenario, to evaluate the security:

$$C_{rs} = \binom{r}{n}\left(\sum_{i=\max(\frac{n+k}{2},n-r+t)}^{\min(n,t)} \binom{r-t}{n-i}\binom{t}{i}\right)^{-1} \tag{4.42}$$

where $k \leq n \leq \min(r, 2t - k)$. When there is a large number of chaff points, a brute force search with interpolation would require less computation than with RS decoding, assuming a decoding operation is similar in complexity to an interpolation. The situation gradually shifts as the number of chaff points decreases. When only a small number of chaff points is present, the number of trials required to obtain the secret polynomial actually reduces as we increase the code word size, depicting a

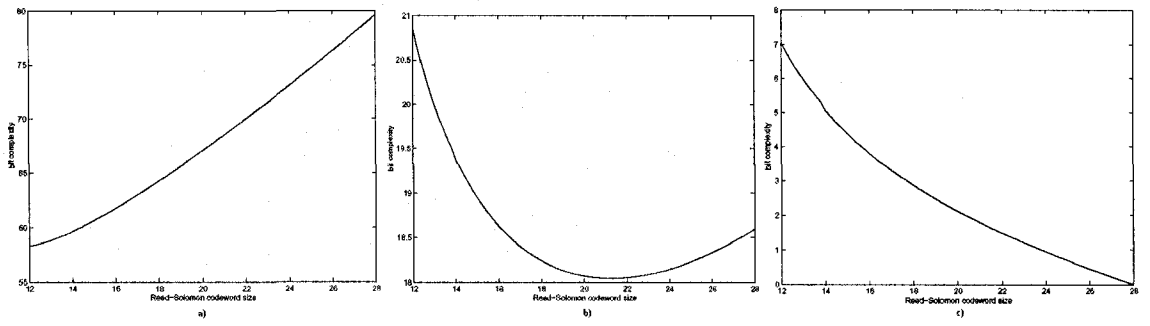

Figure 4.3: Bit complexity, $\log_2(C_{rs})$, with respect to RS code word size, n, for a) a brute force attack on a vault, $(r = 1000, t = 40, k = 12)$, b) an attack on a vault with a significant amount of chaff points identified, $(r = 120, t = 40, k = 12)$, and c) a legitimate decoding operation, $(r = 30, t = 22, k = 12)$[18]

situation similar to a decoding operation by a legitimate user. Figure 4.3 show the complexity of various vault size with respect to the code word size.

In this chapter, we saw that the drastic reduction in information theoretic security and the low performance of the CRC decoder motivates the need for a replacement. By characterizing the Fuzzy vault as a RS code word, we presented two RS decoding algorithms which would work as a decoder for the Fuzzy Vault scheme. Due to the differences in the original approach and the generator polynomial approach to RS codes, some modifications were needed in the Berlekamp-Massey algorithm. We also provided some suggestions on efficient implementations to handle the exceptionally high amount of erasures. Gao's RS decoder was constructed in the view of the original approach and erasures are simply ignored, averting the difficulty with Berlekamp-Massey algorithm, although the comparison of the overall computational cost is unclear. Due to the development of RS decoding algorithms, a reevaluation of the computational security of the scheme was needed. It was found that when the vault size is large, a brute force attack with interpolation outperforms the use of a RS decoding algorithm. However, if an attacker can identify significant amount of chaff points, the situation may shift in favor of a RS decoder.

Chapter 5

Vulnerabilities of the Fuzzy Vault scheme

Although many researchers have provided implementations, few have analyzed the security of the scheme in depth. As various organizations rush to implement biometric technology, it is important that these techniques be shown as secure.

One way to classify the security of a cryptographic scheme is through the amount of information that an attacker is given. The most basic attack is a ciphertext-only attack, which refers to the knowledge of only a set of ciphertexts. The attack is considered successful if the message or the key is revealed. The most common example of a ciphertext-only attack is in systems where users input passwords. It is well-known that these systems are susceptible to dictionary attacks since it is fairly easy to guess user-selected passwords. A more powerful attack is the known-plaintext attack. In this scenario, the attacker have access to a set of ciphertexts with matching plaintexts. The attack is successful if the key is revealed.

One can easily show that Fuzzy Vault algorithm is vulnerable to known plaintext attacks. If an attacker can gain access to the message of a vault, the locking set (i.e. the key) can be easily obtained by verifying the message polynomial on the points in the vault. The following algorithm illustrates such an attack:

Given a vault V_A locked under the set $A = \{a_i\}_{i=1}^{t}$ with the message $M = \{m_i\}_{i=1}^{k}$, where each vault contains r points, $V_A = \{(x_i, y_i)\}_{i=1}^{r}$ and where $X = \{x_i\}_{i=1}^{r}$, perform the following:

– Create an empty set X'

- Set $Y(x) = m_1 + m_2x + m_3x^2 + \ldots m_kx^{k-1}$
- For $i = 1$ to r
  - If $y_i = Y(x_i)$
    - $X' = X' \cup x_i$
- X' is the locking set A

The sets X' and A are equal because all true points (x_i, y_i) lie on the polynomial $Y(x)$ and all chaff points do not.

Any practical cryptosystem must at least be resistant to ciphertext-only attacks. However, we will show in the following sections that the Fuzzy Vault scheme is vulnerable to these attacks if it is not properly used.

In the following section, we propose a change in the definition of the message polynomial in order to preserve security when partial information on the locking set is available. In section 5.2, we present the collusion attack algorithms and the security implication. Suggestions on defenses against the attack are also proposed.

5.1 Partial information loss

Suggested applications for Fuzzy Vault often involve characteristics or properties related to people or objects (ex: favorite movies, biometric features). In the case of fingerprints, there is a very high probability of finding a minutiae near the core. In the case of movie preference, there may be a very popular movie. Hence, it is valid to assume that an attacker would have partial information on the locking set. In Sudan's original description, the message polynomial is defined as:

$$Y(x) = m_1 + m_2x + m_3x^2 + \ldots m_kx^{k-1} \tag{5.1}$$

where k is the size of the message, q is the field size and m_i are the message symbols. There are q^k possible messages. If an attacker can gain access to $k_{known} < k$ points, the number of candidate messages reduces to $q^{k-k_{known}}$. This is a direct result from the number of points required to specify a polynomial of degree k. When combined with the knowledge of spurious polynomials, the security of the system is further reduced.

The message polynomial for RS codes is designed for correcting errors where security is not of concern. In fact, partial information on the location of errors may even increase the chance of retrieving the original message. However, in the case of Fuzzy Vault, it may be desirable to modify the encoding of the message to that similar to Shamir's secret sharing [17] so that partial information on the key would not leak information on the message:

$$Y'(x) = m + a_1 x + a_2 x^2 + \ldots a_{k-1} x^{k-1} \tag{5.2}$$

where m is the message and a_i are randomly generated. With this formulation, no information about the message is revealed unless $k_{known} \geq k$. However, since $m \equiv \{m_1, m_2, \ldots, m_k\}$, the coefficients in $Y'(x)$ are up to k times larger than $Y(x)$, leading to more expensive computations in the larger field. The already large vault size would also greatly increase. This limits the technique from being used in resource-constrainted applications.

5.2 Collusion attack

We describe another attack on the Fuzzy Vault scheme where the attacker is given access to multiple vaults locked by the same key and where the vault size is non-maximal. An example where such an attack is applicable would be the smartcard-based Fuzzy Vault system described by Clancy [18]. In this system, a typical user would carry multiple smartcards issued by different organizations, but the embedded vaults would all be locked by the same key (the user's fingerprint). Another situation would be when an executive issues private instructions more than once to the same group of people, where the key might be their qualifications. We also analyzed the case where multiple vaults are locked with the same secret message and found that a collusion attack is possible even when the vault is maximal.

We first consider the case where the attacker has access to multiple vaults locked by the same key and where the vault size is non-maximal (i.e $r < q$). We found that the attacker can reduce the number of candidate polynomials by exploiting the following properties:

- The keys used to lock the vaults are the same and 'visible' to the attacker
- The chaff points are generated randomly and are independent of the key

- Since the vault size is non-maximal, the chaff points vary from vault to vault

In the collusion attack algorithm, the goal of the attacker is to identify and remove chaff points, thus, reducing the number of spurious polynomials. The key can even be revealed when a sufficiently large number of vaults are available.

5.2.1 Collusion attack algorithm with $A_i = A$

Given n vaults $\{V_{A,1}, V_{A,2} \ldots V_{A,n}\}$ locked under A, where each vault contains r points, $V_{A,j} = \{(x_{i,j}, y_{i,j})\}_{i=1}^{r}$ and where $X_j = \{x_{i,j}\}_{i=1}^{r}$, perform the following:

- Create an empty set X'
- For $i = 1$ to r
  - If $x_{i,1} \in X_j$ for all j
    - $X' = X' \cup x_{i,1}$
- $V_{A,j}^{eff} = \{(x_{i,j}, y_{i,j}) \in V_{A,j} \,|X_{i,j} \in X'\}$, denoted the j^{th} effective vault
- $r_{eff} = \left|V_{A,j}^{eff}\right| = |X'|$, denoted the effective vault size

In another words, the algorithm searches for x values which appear in all the vaults and retain them as possible elements in the true sets. Points that do not appear in all vaults are identified as chaff points and removed. The effective vault is the result of this sieving process. As the number of vaults increases, more chaff points are identified and the set X' approaches the set A since the set A must appear in all n vaults while the randomly generated chaff points may not. Note that the attack is applicable as long as there are some common attributes used to produce the locking sets. It is not required for all elements in the locking sets to be in common and the use of different fields among the n available vault do not protect against the attack as long as the mapping from the attributes to the locking sets are known.

5.2.2 Collusion attack algorithm with $A_i = A$ and $M_i = M$

If the attacker can obtain two or more vaults locked with the same key and with the same secret message, the key can be revealed even easier using similar ideas as the previous algorithm.

Given n vaults $\{V_{A,1}, V_{A,2} \ldots V_{A,n}\}$ locked under the same set $A = \{a_i\}_{i=1}^{t}$ with the same message $M = \{m_i\}_{i=1}^{k}$, where each vault contains r points, $V_{A,j} = \{(x_{i,j}, y_{i,j})\}_{i=1}^{r}$, perform the following:

- Create an empty set V_{eff}
- For $i = 1$ to r
  - If $(x_{i,1}, y_{i,1}) \in V_{A,j}$ for all j
    - $V_{eff} = V_{eff} \cup (x_{i,1}, y_{i,1})$
- $M' = Decode\,(V_{eff})$
- V_{eff} is the effective vault
- $r_{eff} = |V_{eff}|$ is the effective vault size

Since $x_{i,j}$ and $y_{i,j}$ are related by a common polynomial $Y\,(x) = m_1 + m_2 x + m_3 x^2 + \ldots m_k x^{k-1}$, these points must be identical across all vaults. If the chaff points are generated randomly, the probability of two chaff points from different vaults being identical is very low, assuming a reasonably large field size. Hence, the attacker will likely be able to identify A even in the case where $n = 2$. Note that, unlike the previous case, the attacker would succeed even when the vault size is maximal. When the maximum vault size is used, the chaff points in all the vaults contain the same x_i values, but the y_i values they are matched with are randomized while the true points have fixed x_i and y_i. Hence, an attacker can exploit this property to identify chaff points.

Both algorithms assume that all n vaults are locked by the exact same key. However, when using Fuzzy Vault for biometric applications, the keys used to lock the vaults will likely not be identical, but overlapped significantly. In this case, the algorithms can be modified such that $x_{i,j}$ or $(x_{i,j}, y_{i,j})$ appearing in the majority of the vaults are retained. In Clancy's smart-card implementation [18], a minimum distance is also used to avoid confusion between neighbouring $x_{i,j}$. Then, we simply adapt by considering $x_{i,j}$ and $x_{i,j'}$ to be the same if they are within the minimum distance of each other.

5.2.3 Collusion attack algorithm with $M_i = M$

Finally, we consider the case where the attacker is given n vaults locking the same message, but with different keys. If the probability distribution of the locking sets is well defined, then the probability of a particular point being a true point is fixed. We define the probability of a point x being in the locking set as p_t. Then, with probability p_t, the point $(x, Y(x))$ would appear in the vault. When x is not in the locking set, the probability of it being chosen as a chaff point is $p_{chaff} = c/s$, where c is the number of chaff points to be added and s is the number of possible values for a chaff point. Then, with probability $(1 - p_t)p_{chaff}$, we would have (x, y') in the vault, where y' is uniformly distributed over $F_q \backslash \{Y(x)\}$. For clarity, we will assume the vault is maximal, i.e. $p_{chaff} = 1$. Hence, $P(x, y' = y) = (1 - p_t)/(q - 1)$, where y is a value other than $Y(x)$. For $p_t \gg 1/q$, we can see that $p_t \gg (1 - p_t)/(q - 1)$, i.e. $P(x, y' = Y(x)) \gg P(x, y' = y)$. The analysis follows similarly when $p_{chaff} \neq 1$.

Assuming there are $N \gg k$ points where $p_t \gg 1/q$, a collusion attack would proceed as follows:

Given n vaults $\{V_1, V_2 \ldots V_n\}$ with the same message $M = \{m_i\}_{i=1}^{k}$, where each vault contains r points, $V_j = \{(x_{i,j}, y_{i,j})\}_{i=1}^{r}$, perform the following:

- Create a q by q table $Count(x, y)$ and initialize all values to 0.
- For $j = 1$ to n
  - For $i = 1$ to r
    - $Count(x_{i,j}, y_{i,j}) = Count(x_{i,j}, y_{i,j}) + 1$
- $Count(x, y)$ contains the distribution of data over the n vaults
- Create an empty set V_t
- For each $a \in \mathbb{F}$
  - Find the mean u_a of $Count(a, y)$
  - The estimated true point is (a, b) where $b = y'$ where $|Count(a, y') - u_a|$ is maximal
  - $V_t \cup V_t(a, b)$

- $Q(a) = |Count(a, y') - u_a|^2$

- V_t is the estimated true vault, which contains the points most likely to lie on the message polynomial.

- $Q(a)$ is a measure of the quality of the point retained.

- $V_{eff} = m$ pairs of (a, b) with the highest $Q(a)$

- $M' = Decode(V_{eff})$

If the probability of a point being true is high, it will be consistently paired with a particular y value. The algorithm exploits this property to identify a set of candidate true points. When there are a large number of vaults available, the points where $p_t \ll 1/q$ may also be useful as they will appear as to never be paired with a particular y value. To identify which estimated true points are most likely correct, we simply note that the frequency of the ones which appear much greater or much less than its peers. For example, if we see $(1,5), (1,5), (1,7), (1,5), (1,5)$ across five vaults, it would be highly probable that $(1,5)$ is a true point. If we see $(5,3), (5,4), (5,8), (5,8), (5,2)$ across five vaults, the point $(5,8)$ only appears slightly more often than its peers and, thus, is a less reliable candidate for a true point.

In both cases where $M_i = M$, the situation becomes more complex when the field size, q, differs from vault to vault. Although a direct comparison may not be possible, one can likely still narrow down the candidate true points by the relationship: $m(x) = y_i \mod q_i = y_j \mod q_j$.

5.2.4 Security

In the algorithms from section 5.2.1 and 5.2.2, the goal is to identify and remove chaff points to produce a smaller vault, V_{eff}, which still contains all the true points. Hence, we can think of the security of the system when n vaults are used to be equivalent to that of a single vault V_{eff}. Recall from the security analysis of section 2.1 that there are at least $\frac{\mu}{3} q^{k-t} \left(\frac{r}{t}\right)^t$ spurious polynomials in a vault, with probability $1 - \mu$. When generating V_{eff}, the only parameter that was changed in favour of the attacker is the vault size, r_{eff}. The incurred security loss is, thus, $\left(\frac{r}{r_{eff}}\right)^t$.

To better understand the degree at which the security deteriorates, we need to examine the rate at which r_{eff} decreases with respect to the number of vaults available, n, and the vault size, r.

Availability of two vaults locked with the same key is the most likely situation to occur in practise, where the key may, for example, be a person's fingerprint. If we have two vaults locked with the same key and different messages with t true points and $r-t$ random chaff points, all the true points must appear in both vaults while the probability that x of the chaff points match between them is given by

$$P(x) = \frac{\binom{c_1}{x}\binom{s-c_1}{c_2-x}}{\binom{s}{c_2}} \quad \text{where} \quad s = q - t, \quad c_1 = r_1 - t, \quad c_2 = r_2 - t, \tag{5.3}$$

in which s represents the chaff space, and c_1 and c_2 are the number of chaff points in the first and second vault. If we are given $i+1$ vaults with the same parameters, the probability that x_{i+1} chaff points match among all these vaults is given by the following iterative function:

$$P(x_{i+1}) = \sum_{x_i=1}^{c} P(x_i)\frac{\binom{c}{x_{i+1}}\binom{s-c}{x_i-x_{i+1}}}{\binom{s}{x_i}} \quad \text{where} \quad P(x_1) = \begin{cases} 1 & x_1 = c \\ 0 & else \end{cases} \tag{5.4}$$

The expected effective vault size of $i+1$ vault with the parameter set (r, t, q) is thus the sum of the number of true points and the expected number of matches across the vaults:

$$E\{r_{eff}\} = t + \sum_{x_{i+1}=1}^{c} x_{i+1}P(x_{i+1}) \tag{5.5}$$

The variance of r_{eff} can be used as a measure of the accuracy of the mean value.

When $s-c$ is small and s is large, the rate at which $E\{r_{eff}\}$ decreases with respect to n is almost linear, approximately $(s-c)$/vault. This is intuitive since two sets containing almost the entirety of the universe will surely overlap significantly. The missing elements from either set will contain a small portion of the universe and, when chosen randomly, will almost surely be different. However, when $s-c$ is large, $E\{r_{eff}\}$ decreases exponentially with respect to n. From the equation, we can see that the term $\binom{s-c}{x_i-x_{i+1}} = 0$ for all $x_i - x_{i+1} > s-c$. Hence, the effective vault size can at most decrease by $s-c$. In short, the parameters s and c allow us to control the rate at which

Table 5.1: Expected effective vault size and corresponding security given n vaults with the same locking set

$c = 200$				$c = 180$			
n	$E\{r_{eff}\}$	N_{sp}	Bits of security	n	$E\{r_{eff}\}$	N_{sp}	Bits of security
1	220	$5.22 * 10^8$	28.96	1	200	$7.76 * 10^7$	26.21
2	189	$2.50 * 10^7$	24.58	2	157	$6.13 * 10^5$	19.23
3	164	$1.47 * 10^6$	20.48	3	125	6420	12.65
4	142	82241	16.33	4	100	74	6.2
5	123	4650	12.18	5	81	1	0
6	107	286	8.16	6	66	0	-
7	94	21	4.42	7	55	0	-
8	83	2	1	8	47	0	-
9	73	0	-	9	40	0	-

the effective vault size decreases.

Table 5.1 shows the deterioration of security in terms of the number of spurious polynomials, N_{sp}, given n vaults with the same locking set for the following parameters: $q = 2^8, t = 20, k = 16, \mu = 0.01$. The two cases shown, $c = 200$ and $c = 180$, highlight the effect of the number of chaff points on the rate at which security decreases. It can be verified that $P(|r_{eff} - E\{r_{eff}\}| < 8) > 90\%$ for $n \leq 10$ in both cases.

If two vaults contain the same message and are locked with the same key, the situation becomes more alarming. Since the true points are guaranteed to appear in both vaults, we cannot mistake a true point as a chaff point. The only source of error would be in identifying a chaff point as a true point, which would occur when the randomly generated x and y values match. The probability of a match in the y values for two chaff points is $1/(q-1)$. Thus, using the previous equation for $P(x)$ to evaluate the probability of matching x values, we find that the probability of k chaff points matching between two vaults is

$$P(k) = \sum_{x=k}^{c} P(x) \left(\frac{1}{q-1}\right)^k \left(1 - \frac{1}{q-1}\right)^{x-k} = \left(\frac{1}{q-1}\right)^k \sum_{x=k}^{c} P(x) \left(1 - \frac{1}{q-1}\right)^{x-k}. \tag{5.6}$$

For q large and $k > 0$, $(\frac{1}{q-1})^k$ is very small and $(1 - \frac{1}{q-1})^{x-k} \approx 1$. Thus, $P(k) \approx (\frac{1}{q-1})^k \sum P(x)$. Hence, with very high probability, we would have no matches ($k = 0$)

between the two vaults.

The algorithm described in section 5.2.3 exploits the possibility that the locking set is not uniformly distributed. This is certainly true for biometrics. In fingerprints, minutiae locations are most prominent near the core and rarely located at the edges. The question of how many vaults would be needed to compromise the scheme depends heavily on the distribution of the locking sets and the degree of the message polynomial. If the true point has a much higher probability of appearing in the vault than its peers, fewer vaults will be needed to identify it. Similarly, if the degree of the message polynomial is small, the number of reliable true points required to find the message will be small.

Example of a collusion attack

Uludag [19] implemented the Fuzzy Vault scheme using fingerprints. To produce the locking set, the minutiae coordinates of each fingerprint image is first quantized such that they lie in a square tessellation of 7 pixels width. The coordinates are then expressed in 8-bit values such that the range of x or y would be $[0, 255]$. The locking set is then defined to be the set $A = \{a_i\}_{i=1}^{t}$ where $a_i = x_i|y_i$, over the field $GF(2^{16})$. The number of true points, t, is selected to be 18 and the number of added chaff points is 200.

Assuming that an attacker has access to two vaults from the same user, we can estimate the effective vault size using the previously found equations, where $s = 2^{16} - 18$, $c = 200$. With almost 90% probability, the two vaults would have at most one chaff point in common. Thus, the key and the message will likely be revealed immediately.

Actually, the number of chaff points per vault is also optimistic. The size of the fingerprint images from the database is approximately $300 * 400$. With a block size of 7-pixel width, this translates to at most $43 * 58$ possible minutiae coordinates. Hence, out of the $2^{16} = 65536$ possible values for $x|y$, only $43 * 58 = 2494(\approx 4\%)$ are valid. Since the chaff points were drawn randomly from $GF(q)$, only 8 chaff points on average would be valid minutiae coordinates. Then, a simple brute force attack would be able to identify the key and the message. Hence, one must be careful and generate chaff points from the key space and not from the field when the size of the key space is less than that of the field.

5.2.5 Modifications to the scheme

There are several possibilities in modifying the scheme to defend against a collusion attack. We concentrate on the case when the same locking set is used to produce multiple vaults and briefly investigate three possible modifications pertaining to each of the exploitable properties discussed at the beginning of section 5.2: a) 'visibility' of key elements, b) Variability of chaff points across vaults and c) non-maximal vault size.

One-way transform of the locking set

The collusion attack is possible partly because the true points don't change and they are immediately accessible as x_i values. Thus, one way to protect against the attack is to generate different sets of x_i values for different vaults even when the locking sets are the same. Obviously, we must also not allow the attacker to retrieve the locking set elements from x_i as he would then simply reverse the operation and carry out the attack.

The idea is similar to that of cancellable biometrics [12] or password salting. We apply a keyed one-way transform on the locking set for each vault. By the assumption that the attacker cannot reverse the one-way transform, he cannot retrieve x_i. In another word, we compute $a'_i = F(a_i, S)$ for all $a_i \in A$, where $F(x)$ is a one-way function such as a hash function and S is the salt consisting of a random bit string. Then, we proceed with the Fuzzy Vault scheme with $a'_i \in A'$. Since a different S is used for different vaults, the true sets will vary and, thus, the collusion attack would fail.

When the key space is large, one can simply store S along with the vault since it does not help the attacker in retrieving the set A. By using this technique, the security achieved is thus that of computing $F^{-1}(x)$ and the task of identifying the correct polynomial in a vault of size r_{eff} provided that it's less than the security of a single vault.

When the key space is small, the one-way property of $F(x)$ becomes invalid, as a brute force attack would easily provide a complete mapping for all possible a_i. Unfortunately, the entropy of most biometric features tends to be small. A one-way transform of these feature points would provide minimal security enhancement. One possible

solution is to use S as part of the key (e.g. PIN number.) In another word, we increase the key space via the use of a consistent password. The security gained is simply the number of bits in S.

Deterministic chaff point generation

Rather than varying the true points in different vaults, we can also fix the chaff points so that different vaults locked by the same key would have the same set of x_i values. In another word, we generate the chaff points dependent on the locking set A. To do so, we need to produce a consistent bit string from the locking set A. The bit string is then used to initialize a pseudo-random generator, resulting in a consistent set of chaff points. We note that the problem of producing a consistent bit string from A is similar to that of retrieving the message from a potentially corrupt codeword. Hence, we propose the use of error correction codes. The locking set A can be considered a codeword with some symbol errors. By choosing the appropriate parameters, we can perform $Decode'(A)$ to generate a consistent A'. We then initialize a pseudo-random generator to A' and generate the chaff points accordingly.

Note that the size of A' will necessarily be smaller than A. If the size of A is not large enough or it requires a significant amount of error tolerance, the size of A' may become small enough to allow an attacker to perform a brute force attack. By running through all the possibilities for A', he would be able to match a set of chaff points to a set $A = Encode'(A')$. Since a vault consists almost entirely of chaff points, he would then be able to deduce the locking set A.

Maximal vault size

By using the maximal vault size, the vault will contain all the possible x_i values regardless of the locking set. Hence, the chaff points do not vary. As long as the same message is not used twice with the same locking set, we can ensure that a collusion attack would not succeed.

In this chapter, we described an attack on the Fuzzy Vault scheme when more than one vault is available to an attacker. The attack was possible due to the locking set being fixed and in plain form and the variability in the chaff set. It was shown that the attack can rapidly reduce the information security with each additional vault available.

Some suggestions such as salting and the introduction of a PIN were proposed to protect against the attack.

Chapter 6

Conclusion and Future Work

In this thesis, we analyzed the Fuzzy Vault scheme and proposed modifications to improve its security, practicality and privacy.

In order to use fingerprints to produce a Fuzzy vault, various techniques were proposed using minutiae as the locking set. We clarified their security implications and proposed modifications to increase the entropy of the locking set and, consequently, the security of the system.

We also proposed two RS decoding algorithms as a replacement for the CRC decoder. The RS decoding algorithms enjoy much greater reliability, efficiency and security over its predecessor. Our implementation of the first RS decoder for Fuzzy Vault was able to perform the decoding operation in less than a second on a machine with ordinary computational power, without loss in information theoretic security. Implication of the decoding algorithms on computational security was also discussed.

During our study, we also identified some vulnerabilities that need to be addressed. In particular, the collusion attack was found to be applicable to all existing implementations and can quickly compromise the security of the system. We described the attack algorithms for three different situations and proposed modifications to defend against the attack.

6.1 Future work

Our work represents merely a small step towards the development of a practical, secure and privacy-protected biometric authentication system. Many open problems remain

such as the optimal choice of the error correction code used (a subject which has been slightly investigated for the Fuzzy Commitment scheme [16]), the development of a cryptosystem which retains the benefits of Fuzzy Vault with a more manageable ciphertext size, and the suitability of different biometrics (e.g. Iris, facial features, voice.) In the following sections, we'll describe two of the problems that we have began working on.

6.1.1 The case for soft decision decoding for Fuzzy Vault

In communications, the goal of the receiver is to determine the information that was sent from the received analog signal, S_r. In a binary system where 0 and 1 are the two possible signals, a detector may decide that any value greater than 0.5 be assigned to 1 otherwise it is assigned to 0. The detected bit stream is then passed to the decoder for error correction. This process is called hard decision decoding. Both RS decoding algorithms presented in chapter 4 are hard decision decoders. Although simpler in construction, this method discarded important information when quantizing the received analog signal in the detector, information that could be used to better determine the transmitted data. In the same binary system as before, consider the quantization to four levels instead of two: a) $S_r \geq 0.75 \Rightarrow 1$, b) $0.75 > S_r \geq 0.5 \Rightarrow 1^-$, c) $0.5 > S_r \geq 0.25 \Rightarrow 0^-$ and d) $0.25 > S_r \Rightarrow 0$ representing respectively *almost surely* 1, *probably* 1, *probably* 0, *almost surely* 0. The decoder would now be better equiped in determining the message by putting more weights on symbols that are certain and less on uncertain symbols. Figure 6.1 shows the hard and soft decoding of a repetition code, where a bit is repeated 5 times (i.e. 11111 is sent when the message is 1.) In the example, 3 bits were received slightly below the hard decoding threshold and were decoded as 0. However, it is far more probable to receive three *probably* 0's when 1's were sent than to receive two clean 1's when 0's were sent. The question of how to quantize the data and the decision rule to determine the decoded symbol depend on the characteristics of the communication channel and the probability of the symbols being sent. Previous analysis has all assumed that 1's and 0's are sent with equal probability through a white gaussian noise channel.

When constructing a Fuzzy Vault using fingerprints, all existing methods use some form of quantization to account for the effect of noise. In Uludag's implementations, the

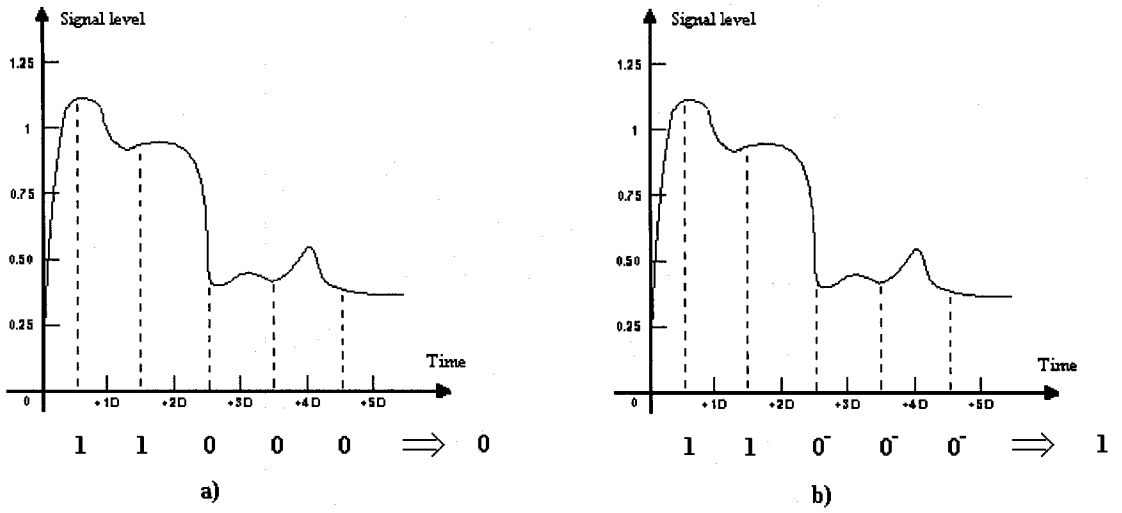

Figure 6.1: a) Hard decision decoding and b) Soft decision decoding of a repetition code

minutiae coordinates are simply mapped to a squared lattice and the corresponding pairs in the vault are retained. Using Clancy's method, minutiae coordinates are compared against (x, y) pairs in the vault and the closest ones are retrieved. In each case, information on the probability that a correct match has occurred were ignored. Assuming that individual minutiae coordinates in different impressions of the same fingerprint follow a Gaussian distribution, then the most obvious measure of how likely it is that a minutia correspond to (x, y) in the vault would be its distance from it. In fact, many other information such as the local quality of the fingerprint image, the location of the minutia and the number of neigboring minutiae can also allow us to provide a better evaluation of the likelihood that the point in the vault corresponds to the detected minutia. As an example, we'll consider Clancy's method of generating the vault, which assures that all points are at least a distance d apart. In this situation, we can view the vault as a signal constellation. If the minutia coordinate is close to the point in the vault being matched to, then we can say that it is almost certain that the minutia is matched to the correct (x,y) pair. The certainty decreases as they become further apart. Figure 6.2 shows the analogous situation of soft decoding in Fuzzy Vault. In the example, the centers of the circles of radius d represent the (x, y) elements in the vault and the crosses represent elements of the unlocking set.

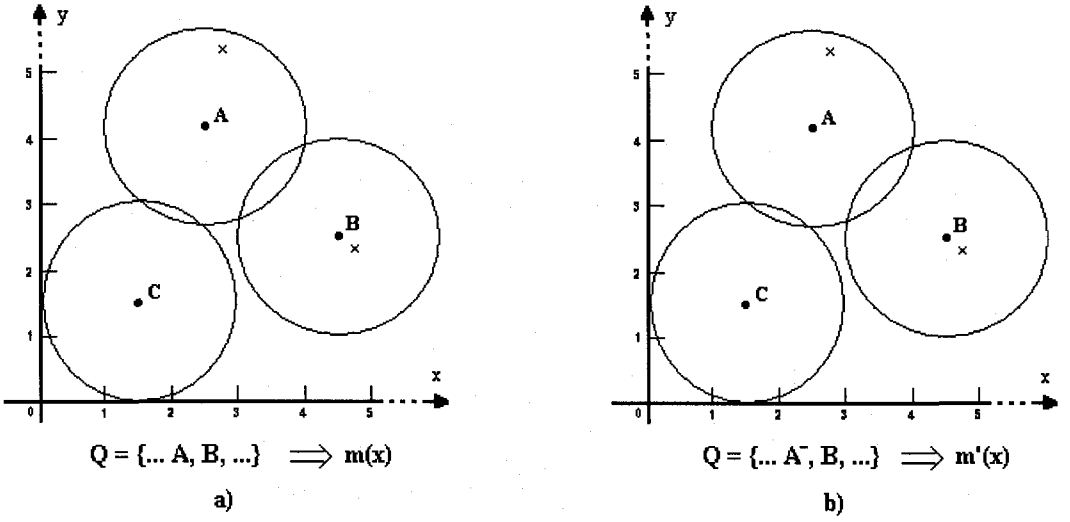

Figure 6.2: a) Hard decision decoding and b) Soft decision decoding of Fuzzy Vault based on the distance from vault elements

We believe that using soft-decoding to retrieve the secret would lead to a reasonable increase in GAR with at most a small sacrifice in FAR. The assumption is that the cause of a legitimate user being rejected is largely due to the false minutia in the unlocking set and that an illegitimate user being accepted is due to his minutia set being close to the locking set by coincidence. To see this, let's first assume that the distribution of the locking/unlocking set is uniform and that different impressions of the same finger produce minutiae sets that follow the Gaussian distribution centered at the minutiae. Figure 6.4 shows a sample outcome of the minutia positions from a legitimate user versus random users. The inner circle englobes the region where the probability of the minutia from the legitimate user would appear is 70.7%, i.e. $\int_A P_{min}(x, y) = 0.707$. The result clearly shows that, by placing more weight on minutiae that are close to the (x,y) pair in the vault would benefit the legitimate user. Note that minutia coordinates from random users can also be used as a model for false minutiae from a legitimate user. Since the probability of false minutiae being identified as less reliable is higher than true minutiae, their effect would be reduced.

The implementation of a soft decision decoder could allow more flexibility in the vault parameters to achieve a particular GAR/FAR, increasing usability. Note that this technique does not affect the computation or information theoretic security. An

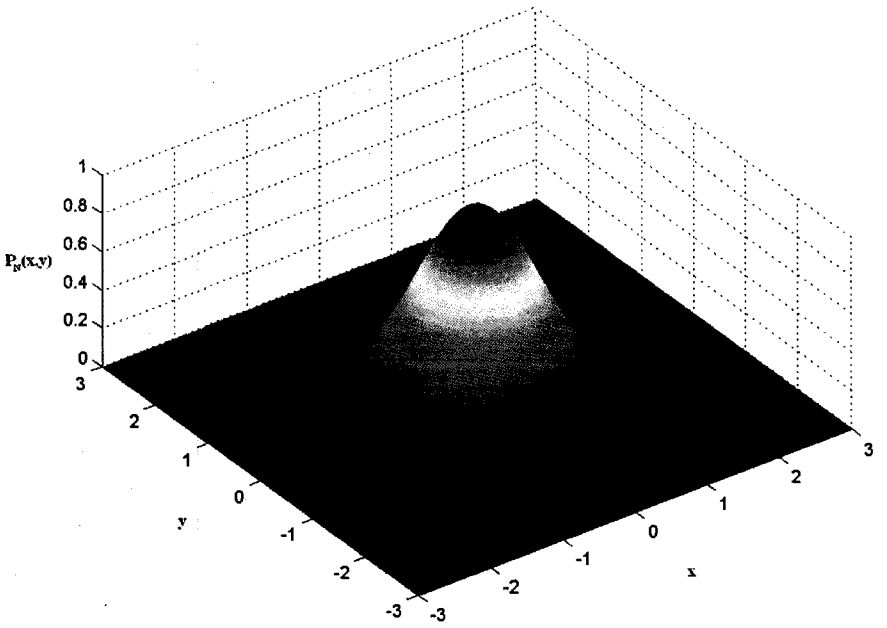

Figure 6.3: 2-D Gaussian distribution centered at the origin

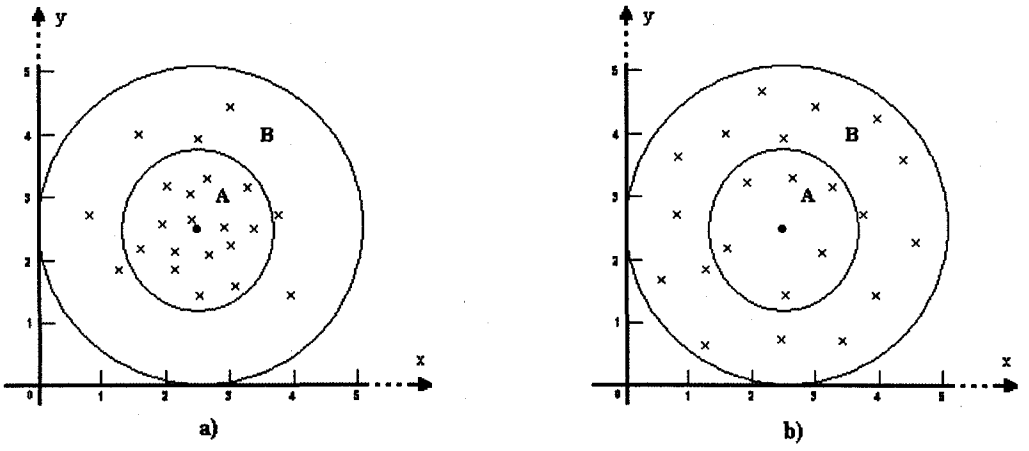

Figure 6.4: a) Minutia coordinates from a legitimate user b) Minutia coordinates from random users with a minutia in the same region

attacker attempting to retrieve the message from a vault knows what the vault elements are and has no reason to pick a unlocking set that does not include the exact points in the vault.

Soft decision RS decoding is a relatively new research area, brought forth by Guruswami and Sudan's list-decoding algorithm[7]. Rather than decoding to a single message, their algorithm can generate a list of code words which are within a certain distance from the received word. By considering the probability of a corrupted code word being correctible rather than the possibility[11], the algorithm effectively correct more errors than all other RS decoding algorithms. This important breakthrough allows for reliable communication over a much longer distance than previously possible. Koetter and Vardy later made improvements by introducing a soft decision decoder for RS codes. Although the complexity of the original Guruswami-Sudan algorithm is much greater than the classic Berlekamp-Massey algortihm, improvements can be observed in ongoing research.

Also, Sudan[8] had noted that a Fuzzy Vault need not be constructed based on RS codes. Perhaps, a different error correction code may lead to a more efficient soft decision decoder.

6.1.2 Privacy in biometric authentication system based on Fuzzy Vault

Although Fuzzy Vault can provide some privacy to its users on its own, the problem of using Fuzzy Vault in a privacy-protected authentication system remains. In section 4.1, we have already discussed the deterioration of information theoretic security in most existing implementations with CRC decoding. Clancy's smart card authentication scheme also does not protect the users' privacy. Its shortfall stems from the use of standard public key signature schemes, which do not rely on information theoretic security. Almost all public key signature scheme work by the assumption that only the legitimate user can provide a valid signature for a message using the private key. The signature verification is carried out by using the public key, which is available to anyone. Although it is computationally difficult to discover the private key from the public key, it is easy to verify whether a given private key is valid. For example, in RSA, a private key, d, can be verified using the relation $m^{ed} = m \bmod n$, true for all

$m < n$, where e and n are public parameters. Also note that the private key is unique for a fixed public key. This commitment between the private key and public key allows the attacker to verify the secret locked in the vault and identify spurious polynomials.

The problem seems to be similar to those in the study of Private Information Retrieval (PIR) and Oblivious Transfer (OT). An investigation into the techniques being used in these fields might yield interesting results. To provide an idea of what we believe should be achieved in order to protect users' privacy, we devised a crude Fuzzy Vault based authentication system, which uses Paillier's probabilistic algorithms [13].

Simply stated, the goal is to protect the locking set in an information theoretic sense, even if it comes at the expense of the security on the secret used to authenticate the user. In another word, the scheme must retain spurious polynomials while safeguarding the secret. One possible approach is to devise a reversible mapping from the secret to numerous polynomials. Since different points lie on different polynomials, these polynomials would appear to be encoded with different locking set and yet all lead to the same secret. The secret can be, as in Clancy's scheme, a specified message signed with a private key in a traditional public key system.

To understand our privacy-protected authentication scheme, some basic notions on the Paillier cryptosystem are needed. Paillier cryptosystem is a probabilistic asymmetric algorithm often used in performing secure computation. Unlike in most public key systems, a message, m, can be encrypted into many possible ciphertexts, $C = E(m_1, r_1)$ with r randomly chosen. An important feature of the system is its homomorphic properties. The product of two ciphertexts will decrypt to the sum of their corresponding plaintexts: $D(E(m_1, r_1) * E(m_2, r_2)) \equiv m_1 + m_2$, and an encrypted plaintext raised to the power of another plaintext will decrypt to the product of the two plaintexts: $D(E(m_1, r_1)^{m_2}) \equiv m_1 m_2$ [13].

We describe a system in which users authenticate to a terminal using a smart card. Each smart card contains a Fuzzy Vault generated using the encoding algorithm described below:

The encoding algorithm for the authentication scheme begins as in section 2.1.1 with $M = E(m_1, r_1)$, where $E(x, y)$ denotes the Paillier encryption algorithm, m_1 is a message signed by the user's private key and r_1 is a random number. Once the true points are added, an equivalent spurious polynomial is computed using $E(m_1, r_2)$,

where $r_2 \neq r_1$, and t random points lying on it are added to the vault. The previous step is repeated h times, with a different r_i each time. Optionally, the scheme can then proceed to add random chaff points as in the original encoding algorithm.

In short, rather than relying on randomly generated spurious polynomials, we purposely insert a number of predefined ones, which are characteristically equivalent to the genuine polynomial but corresponding to different locking set. In another word, the vault contains at least h polynomials of degree k or less which are different representations of the user's encrypted private key. Each of these appears as a spurious polynomial equivalent to the true polynomial, which cannot be verified to be so by any party. They serve to hide the true locking set.

To authenticate the user, the terminal sends a random m_2 to the smart card. If the legitimate user is available, $E(m_1, r_1)$ can be recovered. The smart card then computes $E(m_1, r_1)^{m_2}$ and sends the result back to the terminal. The terminal then performs decryption, $D(E(m_1, r_1)^{m_2}) \equiv m_1 m_2$, and recovers m_1. The user is authenticated if the signed message, m_1, is verified to be authtentic using the user's public key.

Assuming that the exchange between the terminal and the smartcard is secure, this method effectively separates the security of the key in Fuzzy Vault from the key used in authentication through the randomized mapping. The biometrics continue to be protected by information theoretic security while the key used to authenticate the user is protected by computational security in the Paillier cryptosystem or the public key system. However, there are some immediate drawbacks to this technique. By inserting spurious polynomials that lead to the same secret, we effectively allowed the secret to be retrieved using unlocking sets other than that from the legitimate user, increasing the FAR of the system. Also, the number of equivalent spurious polynomials that we can purposely insert is limited by the vault size, n, and the size of the locking set, t. Unless we can generate these polynomials while fixing specific points, each spurious polynomial will require t points in the vault. Hence, we can have at most n/t equivalent spurious polynomials, which may be a fairly small quantity.

Appendix A

Mathematical Background

A.1 Finite fields

A commutative group G is a set of elements on which an operation, $\times$, is defined such that:

1. For $a, b \in G$, $a \times b \in G$
2. For $a, b \in G$, $a \times b = b \times a$
3. For $a, b, c \in G$, $(a \times b) \times c = a \times (b \times c)$
4. For $\forall\ a \in G$, there exists an identity element, $I \in G$, where $a \times I = a$
5. For $\forall\ a \in G$, there exists an inverse element, $a^{-1} \in G$, where $a \times a^{-1} = I$

A field $\mathbb{F}$ is a set of elements on which two operations, $+$ and $\times$, are defined such that:

1. $\mathbb{F}$ defined under $+$ is a commutative group, where 0 or zero is the identity element
2. $\mathbb{F}\backslash\{0\}$ defined under $\times$ is a commutative group
3. For $a, b, c \in \mathbb{F}$, $a \times (b + c) = (a \times b) + (a \times c)$

The number of elements in $\mathbb{F}$ is called the order of the field. A field with finite number of elements is called a finite field or Galois field. A field of order q is often denoted as $\mathrm{GF}(q)$ or $\mathbb{F}_q$. A field $\mathbb{F}_q$ can be extended to a field $\mathbb{F}_{q^m}$ of q^m elements.

The elements of the extended field are then expressed in polynomials of the form $a_{m-1}x^{m-1} + a_{m-2}x^{m-2} \cdots + a_1 x + a_0$, where $a_i \in \mathbb{F}_q$.

A non-zero element $g \in \mathbb{F}_q$ is a generator of the group if the field $\mathbb{F}_q$ can be expressed as $\mathbb{F}_q = \{0, 1, g, g^2, g^3, \cdots, g^{q-2}\}$. An often used result in finite fields is Fermat's little theorem:

$$a^{q-1} = 1 \mod q \tag{A.1}$$

where $a \in \mathbb{F}_q$. Informally, finite fields can be thought of as a finite set of numbers on which operations can be performed such that the same familiar rules in the arithmetic of ordinary numbers hold. A natural advantage of the use of finite fields in practise is in the ability to assure that all elements are representable by a fixed number of bits unlike with numbers in $\mathbb{R}$.

A.2 Lagrange interpolation

Given a set of k points, $\{(x_i, y_i)\}_{i=1}^{k}$, which lie on a polynomial $f(x)$ of degree $k-1$ or less, we may reconstruct the polynomial using Lagrange interpolation:

$$\begin{aligned} \Delta_i &= \prod_{j=1, j\neq i}^{k} \frac{x - x_j}{x_i - x_j} \\ f(x) &= \sum_{i=1}^{k} \Delta_i y_i \end{aligned} \tag{A.2}$$

The solution for $f(x)$ is unique as two polynomials of degree k cannot agree in more than $k-1$ positions.

Appendix B

Mathematical Proofs

B.1 Sums of powers of elements of a field

Theorem B.1.1. *The sum of all the powers of any elements other than unity in* $\mathrm{GF}(q)$ *from* 0 *to* $q-2$ *is zero. For* $a \in \mathrm{GF}(q)$ *and* $a \neq 1$*, the following equation holds:*

$$\sum_{i=0}^{q-2} a^i = 0 \tag{B.1}$$

Proof. Define $x = \sum_{i=0}^{q-2} a^i$. Then, using Fermat's little theorem,

$$ax = \sum_{i=0}^{q-2} a^{i+1} = \sum_{i=1}^{q-1} a^i = \sum_{i=0}^{q-2} a^i = x \tag{B.2}$$

So,

$$ax - x = (a-1)x = 0 \tag{B.3}$$

If $a \neq 1$,

$$x = \sum_{i=0}^{q-2} a^i = 0 \tag{B.4}$$

If $a = 1$,

$$\sum_{i=0}^{q-2} a^i = \sum_{i=0}^{q-2} 1 = q-1 \tag{B.5}$$

□

B.2 Equivalence of the two approaches: proof of the converse

In section 4.2.3, a (n, k) RS code word generated from the original approach was shown to be a valid code word in the generator polynomial approach. To complete the proof, we need to show that every RS code word generated from the generator polynomial approach is also a code word in the original approach. We first show that the code word corresponding to each message is unique in both cases.

Using the original approach, a message, $m(x) = m_0+m_1x+m_2x^2+\ldots+m_{k-1}x^{k-1}$, is mapped to a code word, $C = \{m(1), m(a), \ldots, m(a^{q-2})\}$. Assume a different message, $m'(x)$, also maps to C, then, $m'(x) = m(x)|x \in \mathrm{GF}(q)\backslash\{0\}$. However, two polynomials of degree k can not agree in more than $k-1$ locations. Since $k < q-1$, the assumption must be false. Since there are q^k different messages, there are q^k different code words.

Using the generator polynomial approach, a message, $m(x) = m_0 + m_1x + m_2x^2 + \ldots + m_{k-1}x^{k-1}$, is mapped to a code word, $c(x) = m(x)g(x)$. Assume a different message, $m'(x)$, also maps to $c(x)$, then, $m'(x)g(x) = m(x)g(x)$, which clearly contradicts the assumption. Hence, each message is mapped to a unique code word. As in previous case, there are q^k different code words.

Since every code word in the original approach is also a code word in the generator approach and their code word spaces have the same size, the two set of code words must be identical. □

Bibliography

[1] A. Adler. Vulnerabilities in biometric encryption systems. In *Audio- and Video-based Biometric Person Authentication*, 2005.

[2] Arathi Arakala and Jason Jeffers. Minutiae-based structures for a fuzzy vault. Biometric Consortium 2006 Conference, 2006.

[3] Ren Shen Ee-Chien Chang and Francis Weijian Teo. Finding the original point set hidden among chaff. In *ACM Symposium on Information, computer and communications security*, 2006.

[4] S. Fedorenko. A simple algorithm for decoding Reed-Solomon codes and its relation to the Welch-Berlekamp algorithm. *IEEE Transactions on Information Theory*, 3:1196–1198, 2005.

[5] Shuhong Gao. *Communications, Information and Network Security*, chapter A new algorithm for decoding Reed-Solomon codes, pages 55–68. Kluwer Academic Publishers, 2003.

[6] D. Gorenstein and N. Zierler. A class of cyclic linear, error-correcting codes in p^m symbols. *J.SIAM*, 9:207–214, 1961.

[7] V. Guruswami and M. Sudan. Improved decoding of Reed-Solomon and algebraic-geometric codes. *J.SIAM*, 9:207–214, 1998.

[8] A. Juels and M. Sudan. A fuzzy vault scheme. In *IEEE International Symposium on Information Theory*, page 408, 2002.

[9] A. Juels and M. Wattenberg. A fuzzy commitment scheme. In *The Sixth ACM Conference on Computer and Communications Security*, pages 28–36, 1999.

[10] Liu Zhaoqing Li Qiong and Niu Xiamu. Analysis and problems on fuzzy vault scheme. In *International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pages 244–250, 2006.

[11] R. J. McEliece. The GuruswamiSudan Decoding Algorithm for Reed-Solomon Codes. Technical report, The Interplanetary Network, 2003.

[12] J. Connell N. Ratha and R. Bolle. Enhancing security and privacy in biometrics-based authentication systems. *IBM System Journal*, 40(3):614–634, 2001.

[13] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. *Lecture Notes in Computer Science*, 1592:223–238, 1999.

[14] S. Reed and G. Solomon. Polynomial codes over certain finite fields. *J.SIAM*, 8:300–304, 1960.

[15] D. Costello S. Lin. *Error Control Coding.* Prentice Hall, 2003.

[16] D. Schipani and J. Rosenthal. Coding solutions for the secure biometric storage problem. Submitted to International Symposium on Information Theory 2007, 2007.

[17] A. Shamir. How to share a secret. *Communications of the ACM*, pages 612–613, 1979.

[18] N. Kiyavash T. Clancy and D. Lin. Secure smartcard-based fingerprint authentication. In *ACM SIGMM workshop on Biometrics methods and applications*, pages 45–52, 2003.

[19] S. Pankanti U. Uludag and A. Jain. Fuzzy vault for fingerprints. In *the Audio- and Video-based Biometric Person Authentication*, pages 310–319, 2005.

[20] U. Uludag and A. Jain. Securing fingerprint template: Fuzzy vault with helper data. In *IEEE Workshop on Privacy Research In Vision*, page 163, 2006.

[21] U. Uludag and A.K. Jain. Fuzzy fingerprint vault. In *Biometrics: Challenges Arising from Theory to Practice workshop*, pages 13–16, 2004.

[22] Stephen B. Wicker. *Reed-Solomon Codes and their Applications.* IEEE Press, 1994.

[23] S. Yang and I. Verbauwhede. Automatic secure fingerprint verification system based on fuzzy vault scheme. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 609–612, 2005.